

USER GUIDE

CALCULUS

WITH

MAPLE

based on materials from
Department of Mathematics
North Carolina State University

David Royster
UNC Charlotte
droyster@email.uncc.edu
<http://www.math.uncc.edu/~droyster>

This document is an introduction to *Maple 6* that will lead the reader to some understanding of how *Maple* may be used in college calculus classes, especially those from North Carolina State University. These are addressed to teachers of AP Calculus and will compare what is done in *Maple* to what can be done using the graphing calculator. This is not meant to be an exhaustive treatise on *Maple*, but just an introduction.

The highlighting notation we use is as follows:

- We **boldface**, underline and print in blue **key Maple ideas**. For example the concept of a **Maple command line** in a **Maple worksheet** is discussed in section 1.
- We **boldface** and print in red the names of Maple commands. For example, we discuss the **plot** and **with(plots)** commands in section 1.
- We use **boldfacing** and underlining to highlight other key concepts. For example, we discuss the idea of **converting a command line to a text line**.

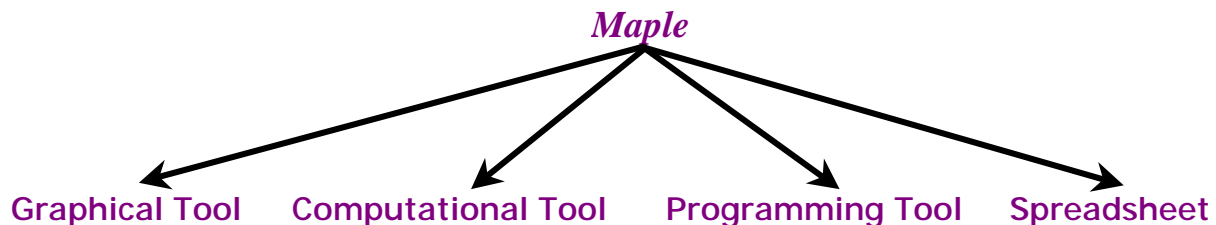
The *Maple* Requirement in Calculus at North Carolina State University

In the NCSU Catalog course listing for the calculus courses MA 141, MA 241 and MA 242, one finds the statement that students taking calculus at NCSU are required to learn to “use computational tools”. At the present this statement means that students in the calculus courses are required to complete the *NCSU Maple Basics* Lessons and Homework.

1. What is *Maple*?

Maple is a package of software tools such as *Microsoft Office2000*, which contains the *Microsoft* programs (1) *Word*, (2) *Excel*, (3) *PowerPoint*, (4) *Explorer* and (5) *Outlook*:

In contrast *Maple* contains (1) a Graphics Tool, (2) a Computation Tool, (3) a Programming Tool, and (4) a Spreadsheet:



In the following pages we will look at some of the different ways *Maple* may be used in the study of calculus.

1.1 *Maple* Basics

The first thing you need to know about *Maple* is that *everything* is centered around the *Maple* worksheet. The worksheet is the canvas upon which you do your work. When you start up the *Maple* program the *Maple* window appears on your screen. Have a look at the sample *Maple* window below. The *Maple* program uses a standard screen format, namely it divides the *Maple* window into two parts, the top menu bar, and the bottom worksheet window. The top menu bar always stays put, while the bottom worksheet window changes whenever you change from one worksheet to another. The worksheet window is where you type in and enter your commands, and it is where *Maple* gives you the results. In this sample we loaded the **plots** package with the command

with(plots):

and then used the **plot3d** command to plot a portion of the

graph of the function $f(x, y) = x^2 + y^2 - 9$. The entire *Maple* program is very large, and

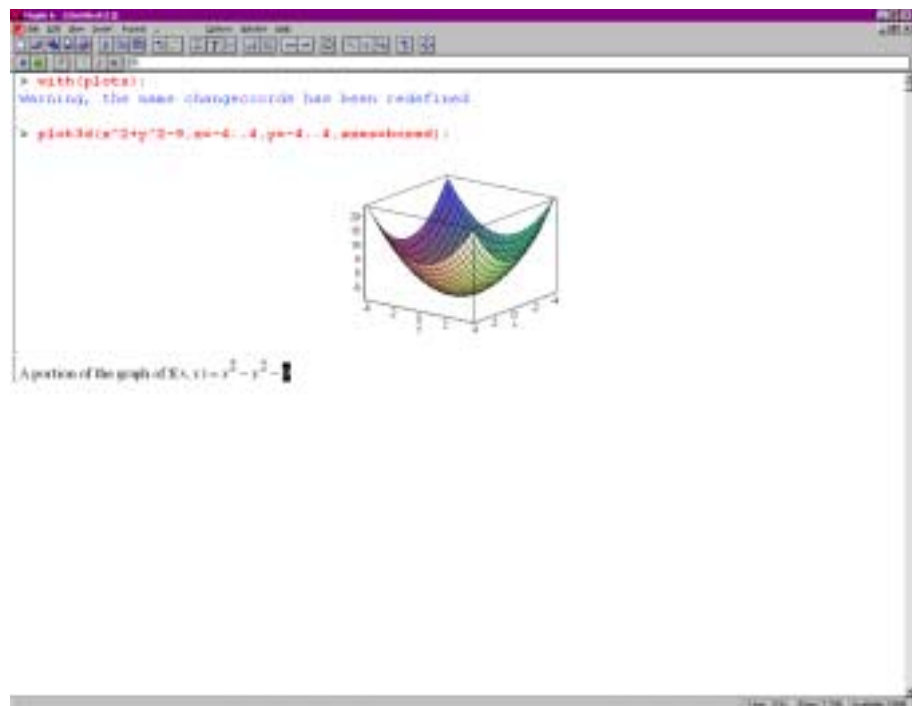


Figure 1. A sample *Maple* worksheet

consequently most of *Maple* is not stored in active memory. *Maple* divides itself into pieces called *packages*, like the *plots* package mentioned above. The following window shows the *Maple* commands that become available once the *plots* package is loaded in memory with the `with(plots)` command.

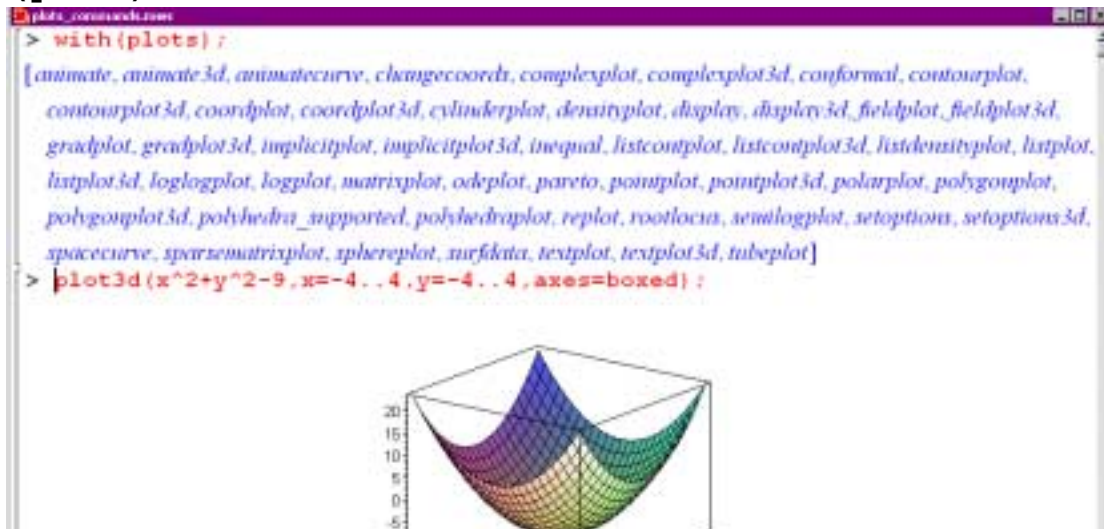


Figure 2. The *Maple* commands in the *plots* package

1.2 Saving a Worksheet

To save the worksheet one would click on File on the top menu bar, and then click on Save as. A dialogue window would open in which you type the name of the file and then click Save. The filename ends with the extension .mws.

1.3 The colon and semicolon in Maple

Let's now return to the sample Maple worksheet and look at the symbol used to end each line.

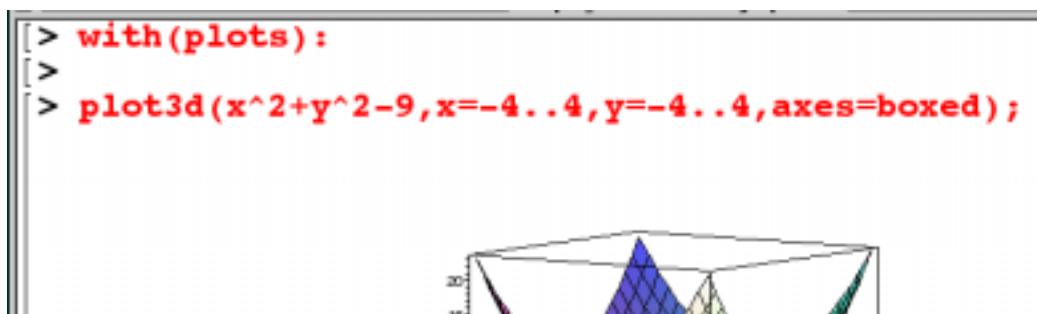


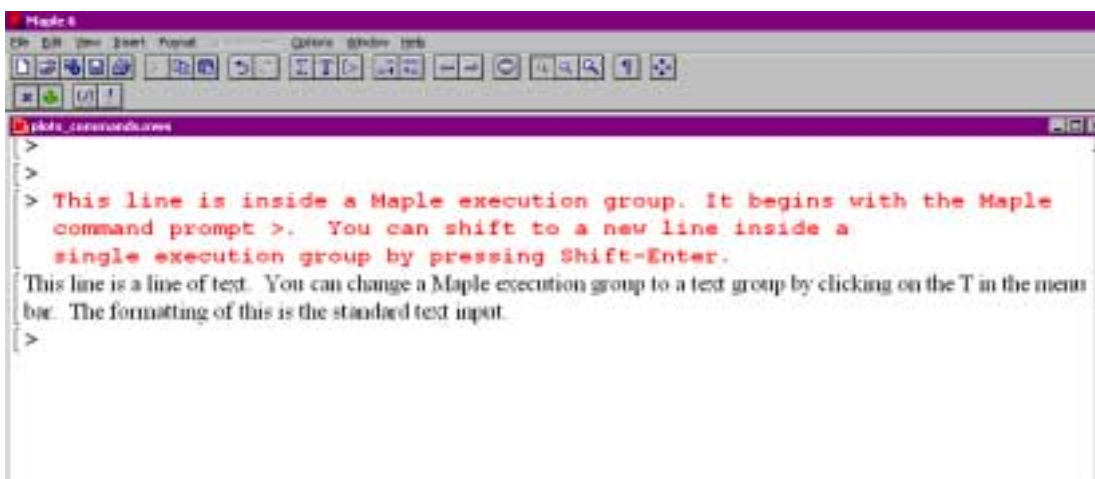
Figure 3. The colon versus the semicolon.

You'll notice that the **first** line ends in a colon, and there is no discernable *Maple* output associated with that command. On the other hand the **third** line in Figure 3 ends with a semicolon, and the output is the graph of the function. Figure 2 also shows the *Maple* output when the command `with(plots)` is followed by a semi- colon. These examples illustrate the general rules on the colon and semicolon.

- All command lines, **except** lines that **begin** with a question mark ? must end with either a colon or semicolon.
- If a command line ends with a colon, then the output of that command is suppressed.
- If a command line ends with a semicolon, then the output of that command is displayed.

1.4 Maple-Inputs and Maple Text

In addition to entering and executing commands in a *Maple* worksheet, one often needs to enter text. This text might be the statement of a problem, or it might be comments you use to explain a calculation in the middle of a worksheet.



The most basic unit in a *Maple* worksheet is the **execution group**. One tells *Maple* what to do by entering commands in a *Maple* execution group. It is bounded on the left by a left bracket [followed by the “greater than” sign >. Thus an execution group composed of just one line begins with the prompt [`>`. There is one such execution group at the bottom of the above picture and two at the top of the picture. The third execution group above consists of three lines inside a single execution group. By default, text typed inside a *Maple* execution group is printed in red.

The text in black in the above picture is inside a text field. No *Maple* commands can be issued from within a text field. When you need to make comments in a *Maple* worksheet you can switch an execution group to a text field as explained in the above picture, and then enter text. Or you can begin a line inside an execution group with the # symbol, which tells *Maple* that the following is a comment, and *Maple* should simply ignore it.

1.5 Getting HELP

The *Maple* package provides an extensive HELP facility that all *Maple* users become acquainted with soon after beginning to experiment inside a *Maple* worksheet. Learning to use the *Maple* program will require practice since you will be learning a new language that has a specific syntax. For example, we have already learned that *Maple* knows that you are issuing a command when a line inside an execution group ends with a colon or semicolon. But what should you do if you happen to forget the syntax for a specific command? The sequence of commands in the worksheet pictured below illustrates a situation that can occur for a new *Maple* user.

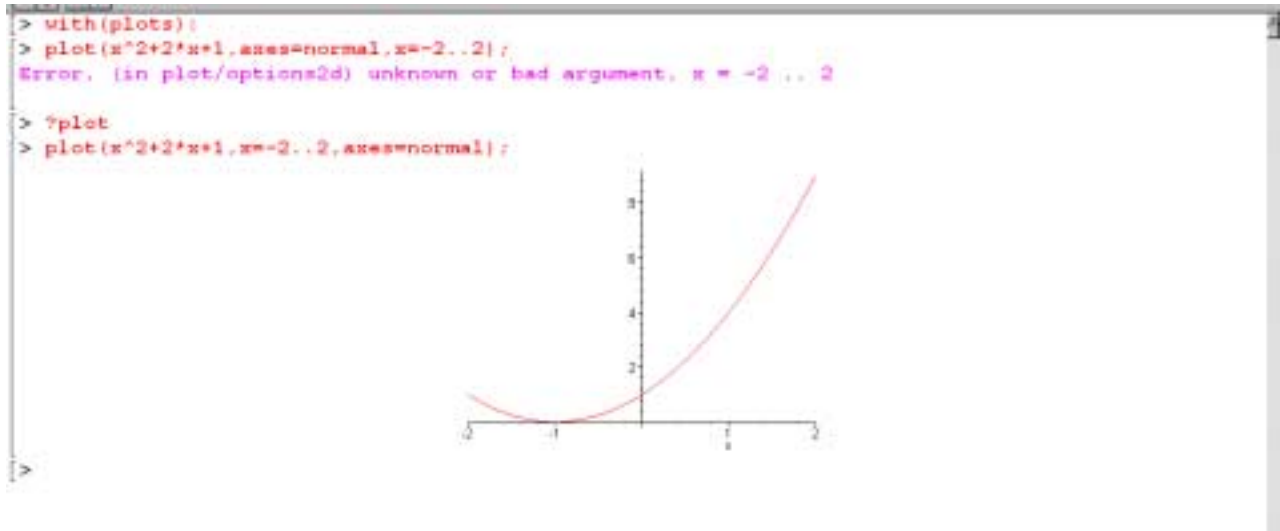


Figure 4. *Maple* Error Messages

After loading the plots package the author of this worksheet tried to plot the graph of the function $f(x) = x^2 + 2x + 1$ on the interval $[-2, 2]$. The author also has chosen to use the **normal** x - y axes plot option for the plot. Unfortunately the author has forgotten the correct order in which to input the information into the **plot** command. This is again a Maple syntax problem. *Maple* has issued, in default color magenta, the following error message:

Error, (in plot/options2d) unknown or bad argument, x = -2..2

In Figure 4 above we observe that the user was not phased by this problem. Since he couldn't remember the correct syntax, he ordered up the help window with the *Maple* command

```
[> ?plot
```

What happened next was that *Maple* opened the help window for the *Maple* command **plot**.

The Help window has the name of the *Maple* command at the top followed by a brief description of the action associated with that command:

plot - create a two-dimensional plot of functions

Next on the page is the list of **Calling sequences**, followed by the list of **Parameters**, which is followed by a **Description** section. The **Calling sequences** tell you the ordering of the inputs, and the **Parameters** tell you the names of the inputs for the command. Notice that there are two calling sequences for the plot command, and this is typical of most *Maple* commands. There are typically two or more forms of the calling sequence for any *Maple* command. The **Description** section describes various aspects of the command.

1.6 The assignment operation

One of the most basic things that you will do over and over again inside *Maple* worksheets is to assign names to various expressions. That is, you will name the objects that you want to manipulate in the worksheet. The way that you notify *Maple* that you are making a name assignment is to use the assignment operator, which is a colon followed by the equality sign. For example, if you want to give the name f to the expression $x^2 - x - 10$ in *Maple* you would make this assignment as shown in Figure 7. The assignment is made in red, and *Maple* replies with the output in blue. On the next command line the name f is entered with the command `> f;` and *Maple* replies with the value of f , namely $x^2 - x - 10$. For as long as you do not delete the active memory *Maple* will remember that the name f has the value $x^2 - x - 10$.

```

> f:=x^2-x-10;
f:=x^2-x-10
> f;
x^2-x-10
> |

```

Figure 7. The expression $x^2 - x - 10$ is assigned to the name f .

2. Maple's Graphical Tool

One of the most useful ways to study functions and equations is to study them visually, and *Maple* contains a powerful graphical tool to plot the graphs of functions of 1 and 2 variables, as well as the graphs of equations in 2 and 3 variables. The plots shown in figures 1 and 2 above are examples of *Maple* plots. The further you progress, the more you will learn about how to use *Maple*'s plotting tools to help you do your work. At times it is simply too time consuming and too difficult (impossible in general) to sketch the graph of a complicated function or equation by hand. By learning to use *Maple* to do your plotting for you, you will greatly extend your ability to study functions and equations. You will be able to get a look at the graph of a function or equation in just a few seconds, namely in the time it takes to enter the appropriate plot commands in a *Maple* worksheet. One uses:

- **plot()** to plot the graph of functions of 1 variable
- **plot3d()** to plot the graph of functions of 2 variables

- **implicitplot()** to plot the graph of an equation in x and y
- **implicitplot3d()** to plot the graph of an equation in x , y and z

The structure of the plot commands are much like those of the graphing calculator, with the exception that all of the window information goes into the command itself. There is more control over the output of the graph, but it all has to be given in one line. In all of the GUI platforms once a plot is done, there is a window where you can make some changes in the format of the plot.

3. Maple's Computational Tool

Maple is well known for its ability to perform a wide variety of complicated mathematical calculations and manipulations. In the calculus sequence of courses you will learn to use Maple to do all your basic calculations, after you master the conceptual basis for each calculation. You will use Maple to:

- compute limits of functions
- compute ordinary and partial derivatives of functions
- solve systems of algebraic equations
- work with lines, planes, surfaces and other geometrical objects
- do calculations with vectors and matrices
- compute anti-derivatives
- compute integrals: Riemann, line, surface, surface area and flux integrals
-

Maple organizes its computational tools into [packages](#). You have already seen above in **Figure 2** the list of commands in the [plots package](#). Below we show the commands in the commonly used **student**, **linalg** (linear algebra) and **DETools** packages.

```

some maple_packages.mws
> with(student);
[D, Diff, Doubleint, Int, Limit, Lineint, Product, Sum, Tripleint, changevar, combine, completesquare, distance, equate,
extrema, integrand, intercept, intparts, isolate, leftbox, leftsum, makexproc, maximize, middlebox, middlesum, midpoint,
minimize, powtabs, rightbox, rightsum, showtangent, simpson, slope, summand, trapezoid, value]

> with(linalg);
[BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian, addcol, addrow, adj, adjoint, angle,
argument, backsub, band, basis, bezout, blockmatrix, charmat, charpoly, cholesky, col, coldim, colspace, colspan,
companion, concat, cond, copyinto, crossprod, curl, definite, delcols, delrows, det, diag, diverge, dotprod, eigenvals,
eigenvalues, eigenvectors, eigenvects, entermatrix, equal, exponential, extend, ffgausselim, fibonacci, forwardsub,
frobenez, gauzselim, gauzsjord, genegns, genmatrix, grad, hadamard, hermite, hessian, hilbert, ltranspose, ihermite,
indexfunc, innerprod, intbasis, inverse, ismith, isimilar, iszero, jacobian, jordan, kernel, laplacian, leastsqrs, linsolve,
matadd, matrix, minor, mibpoly, mulcol, mulrow, multiply, norm, normalize, nullspace, orthog, permanent, pivot,
potential, randmatrix, randvector, rank, ratform, row, rowdim, rowspan, rowvec, scalarmul, singularvals, smith,
stackmatrix, submatrix, subvector, subbasis, swapcol, swaprow, sylvester, toeplitz, trace, transpose, vandermonde,
vecpoteur, vectdim, vector, wronskian]

> with(DEtools);
[DEnormal, DEplot, DEplot3d, DEplot_polygon, DFactor, Dchangevar, GCRD, LCLM, PDEchangecoords,
RiemannPsols, abetsol, adjoint, autonomous, bernoullisol, buildsol, buildsym, canoni, chintzol, clairautsol,
constcoeffsols, convertAlg, convertsys, dalembertsol, de2diffop, dfieldplot, diffop2de, eigewing,
endomorphism_charpoly, equinv, eta_k, eulersols, exactsol, expsols, exterior_power, formal_sol, gen_exp, generate_ic,
genhomosol, hamilton_eqs, indicialeq, infgen, integrate_sols, inafactor, kovacicols, leftdivision, liesol, line_int,
linearsol, matrixDE, matrix_riccati, moser_reduce, mult, newton_polygon, odeadvisor, odepde, parametricol,
phaseportrait, poincare, polysols, ratsols, reduceOrder, regular_parts, regularsp, riccati_system, riccatisol,
rightdivision, separablesol, super_reduce, symgen, symmetric_power, symmetric_product, systest, transinv, translate,
natranslate, varparam, zoom]

```

4. Maple as a Programming Language

Maple is itself a modern programming language. You can write programs in a Maple worksheet using standard programming commands coupled with the specific commands available in Maple packages. For example you can write **do-loops** and **if-while-then** statements to do repetitive calculations, and you can create new Maple functions to do specialized calculations in a Maple worksheet. You will begin to explore and use this aspect of Maple in the 3rd MA 241 lesson when you study Euler's one-step method for approximating the solution of an initial value problem. By the time you finish your 3rd calculus course, MA 242, you will be using new procedures developed in the Maple lessons to evaluate surface integrals and flux integrals.

Maple has many of the commands available in standard programming languages. However the syntax and output of these commands may vary from what you are used to, so you should always consult the [help pages](#) before using programming commands. As mentioned earlier, typing

▷ ?if

or

▷ ?do

will launch the help screens for these commands.