A very quick introduction to Maple and MATLAB
Math 32
Version: Sep 25, 2006

This handout discusses how to perform some important basic operations in the software packages Maple and MATLAB, including graphing functions, taking derivatives, and doing integrals. The instructions assume PC's, but Macs and Unix machines should be similar.

Maple vs. MATLAB: The difference between Maple and MATLAB is that Maple is better at symbolic computation and dealing with functions (e.g., calculus), whereas MATLAB is better with data (e.g., engineering). Both systems can do what we need for this class (mostly 3-D graphing), so I would say that if you are majoring in engineering or science, you should learn to work with MATLAB, and if you are majoring in something else, like math, you should learn Maple. Alternately, if you get an account on the math department computers, you can try using both and see which one you prefer.

Some Maple basics

Running Maple. If you own your own copy of Maple, you should have instructions on how to start the program up. If you're using Maple on the math department computers, it's one of the options on the "Start" menu. Once you start Maple and are looking at a worksheet (a place to type commands), you may need to press the `[>` button to get a `>` prompt, depending on your version of Maple.

Basic Maple. Here are some of the basic Maple commands you'll need. We'll assume that you're typing the commands in at the `>` prompt, though if you prefer to use the menu buttons at the top to do the same things, that's fine.

In this course, you can think of Maple as a very sophisticated graphing calculator. For instance:

```
> 47*59*71;
```

$$196883$$

The important thing to notice here is that every Maple command must end with a semicolon. If you omit the semicolon, Maple thinks that you're typing in a very long command and haven't finished yet.

The standard math operators include the usual `+`, `-`, `*`, `/`, `^`, and `sqrt`. Most standard mathematical functions are included (e.g., `exp(x)`, Maple's version of e^x, `sin()`, `ln()`, `arctan()`, etc.), and the number π is represented *exactly* by the constant symbol `Pi`. This brings up another important point, namely, that Maple is designed to work with exact values, and not just with numerical approximations:

```
> arctan(sin(Pi/2));
```

$$\frac{1}{4}\pi$$

To get numerical values, use the command `evalf()`. Beware of roundoff error, though:

```
> evalf(Pi);
```

$$3.141592654$$

```
> sin(%);
```

$$-4.102067615\ 10^{-10}$$

```
> sin(Pi);
```

$$0$$

Any name (sequence of letters and numbers, starting with a letter) that Maple doesn't know already is assumed to be an unspecified mathematical unknown. This can produce some odd-looking effects if you're not prepared, especially if you forget that all multiplication must be explicitly stated with the * operator:

```
> ax*x;
```

$$ax\ x$$

```
> a*x*x;
```

$$a\ x^2$$

```
> mc*mc;
```

$$mc^2$$

Note the subtle difference in the output of the last two examples. In Maple output, mc^2 is definitely not the same as $m\ c^2$.

Names can also be used as variables. Assignment is done with the `:=` operator:

```
> y := (x+3);
```

$$y := x + 3$$

```
> expand(y^3);
```

$$x^3 + 9x^2 + 27x + 27$$

One of Maple's strengths is its ability to do symbolic computations from calculus. For instance:

```
> limit(sin(3*x)/x,x=0);
```

$$3$$

```
> diff(exp(-x),x);
```

$$-\mathbf{e}^{(-x)}$$

```
> int(1/(1+z^2),z);
```

$$\arctan(z)$$

```
> int(1/(1+z^2),z=0..infinity);
```

$$\frac{1}{2}\pi$$

Remember that the only way to get e^x in Maple is to use `exp(x)`; in particular, the only way to get the constant $e = 2.71828\ldots$ is `exp(1)`. Also, even though Maple omits the $+C$ in an indefinite integral, you should still include it when appropriate.

Another of Maple's strengths is solving equations. For instance:

```
> curve := solve(y^2=x^3+x^2,y);
```

$$curve := \sqrt{x+1}\,x, -\sqrt{x+1}\,x$$

Note that we have also stored the solution in the variable `curve`. More precisely, `curve[1]` is now the name of the first solution, and `curve[2]` is the name of the second.

The main way in which we'll use Maple, however, is to make pictures. For instance, the command

```
> plot(x/2+1/2,x=-2..2);
```

draws the graph $y = \frac{x}{2} + \frac{1}{2}$ for $-1 \leq x \leq 2$. You can also draw the graph of the result of a `solve` command; try:

```
> plot({curve[1],curve[2]},x=-2..2);
```

Note that this command uses the results of our previous `solve` command. Also, be careful to use curly braces {} around `curve[1],curve[2]`, and not just parentheses.

It is also useful to know that you can combine several plots into one. First, you have to load the `plots` Maple package:

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

Note that we finished the command with a colon and not a semicolon. This tells Maple to do the command, but to keep quiet about the results. In any case, we next save the results of our plots in variables:

```
> curveplot := plot({curve[1],curve[2]},x=-2..2):

> lineplot := plot(x/2+1/2,x=-2..2):
```

Again, note the colons. The combined plots are displayed with the command:

```
> display({curveplot,lineplot});
```

Note that we go back to using a semicolon, since we want to see the results of this last computation.

Finally, if you want to print out what you've done, pull down the "File" menu and select "Print". *Warning:* this will print out your entire worksheet, that is, everything you've done, mistakes and all. If you're printing out something for an assignment, you can save lots of paper by first deleting everything that you don't want to print. (To do this, just select the unwanted sections with the mouse, and press Delete.)

Try these:

1. Compute the indefinite integral $\int e^{x/a} \sin(-bx)\,dx$, where a and b are arbitrary nonzero constants, and check your answer by finding its derivative with respect to x.

2. Graph the functions $\sin 2x$ and $\cos \sqrt{x}$ $(0 \leq x \leq 2\pi)$ on the same set of axes.

Graphs of functions of 2 variables in 3-space. Here is where Maple really starts to come in handy for Math 32. To plot, for instance, $z = x^2 + y^2$, do:

```
> with(plots):

> plot3d(x^2+y^2,x=-2..2,y=-2..2,axes=normal);
```

Unlike `plot`, `plot3d` requires you to specifically request axes to appear. If you omit the `axes=normal` part of the command, none will appear.

A really cool thing: To see this 3-D plot (or any 3-D plot) from a different viewpoint, click in the picture window with the mouse and drag the picture around. Try it — I hope it will make the rest of this handout seem worthwhile. Note also that while you have a plot selected, you can change various attributes of the plot, such as the axes style, the colors, and so on.

There is also a command `display3d`, which is part of the `plots` package, and is entirely analogous to the `display` command for 2-D plots.

Contour plots. To draw a contour map for a function, do (again, after having entered `with(plots):`):

```
> contourplot(x^2+y^2,x=-2..2,y=-2..2);
```

Try comparing the graph and the contour plot of several multivariable functions $f(x, y)$.

Help! Maple has a very nice menu-driven help system. In particular, for a very complete introduction to Maple and all of its features, select "New User's Tour" under the Help menu and follow the ensuing instructions.

A few basic ideas and commands in MATLAB

Running MATLAB. If you own your own copy of MATLAB, you should have instructions on how to start the program up. If you're using MATLAB on the math department computers, it's one of the options on the "Start" menu.

In MATLAB, you can proceed by typing commands at the `>>` prompt, much like Maple. One important difference is that commands need not be terminated by a semicolon; in fact, if you end a command with a semicolon, the output is not displayed, in the same way that the colon prevents output display in Maple.

Another important difference is that assignment is done by the ordinary `=` and not by `:=`, as we would use in Maple.

Warning. The MATLAB information below is based only on manuals, Google searches, and guesswork; it may need to be modified to work correctly. Please consult the MATLAB menu-driven help system, and check the course webpage for updated versions of this document. Also, please report any errors or problems to me at `hsu@math.sjsu.edu`.

Data and functions in MATLAB. The basic object in MATLAB is a *matrix*, or two-dimensional array. (Matrices are what puts the MAT in MATLAB.) Instead of storing functions and formulas as abstract symbolic ideas, as Maple does, MATLAB generally stores functions and formulas as tables of values. For example, if we want to graph $\sin(x)$ for $-2 \leq x \leq 2$, we store a table of values of x and $y = \sin(x)$ as follows. First, we store the values of x from -2 to 2, increasing by a step of .01 (ie.., -2, -1.99, etc.) in a 1×201 array:

```
>> X = -2:.01:2;
```

Then we store the corresponding values of y in an array Y:

```
>> Y = sin(X);
```

To plot these points, we do:

```
>> plot(X,Y);
```

One tricky point here is that MATLAB assumes by default that operations are matrix operations. So, for example, if we want to graph $y = 3x^2 - 5$ for $-2 \le x \le 2$, when we set Y to be a table of values of $y = 3x^2$, MATLAB will interpret the very natural formula `Y=3*X^2` as attempting to set Y to be 3 times the matrix X multiplied by itself, which is actually undefined. To tell MATLAB to perform operations entry by entry, as we usually require when graphing functions, we use the "entrywise" operations `.*`, `.^`, and `./`, as seen in this example:

```
>> X = -2:.01:2;

>> Y = 3*X.^2;

>> plot(X,Y);
```

Note that if we multiply by a constant (the `3*` in the above example), we do not use the "entrywise" operation.

Dealing with constants in a formula $y = f(x)$ is even trickier. For example, suppose we want to graph $y = -5x^2 + 7$ for $-2 \le x \le 2$. To add that 7, we actually need to create a matrix of all 7's that is the same size as X and Y. The command `ones(size(X))` creates a matrix of all 1's that is the same size as X, so to graph $y = -5x^2 + 7$, we do:

```
>> X = -2:.01:2;

>> Y = -5*X.^2 + 7*ones(size(X));

>> plot(X,Y);
```

Note that `+` is automatically an entrywise operation.

3-D graphs and contour maps. To graph $z = f(x, y)$ over a region $-2 \le x \le 2$, $-3 \le y \le 3$, for example, you first have to set up what is called a *meshgrid* (two arrays of x and y input values):

```
>> [X Y] = meshgrid(-2:.01:2,-3:.01:3);
```

For technical reasons, note that both X and Y will be 201×301 matrices.

Then, for example, to graph the surface $z = x^2 + y^2 - 2$ on this region, we do:

```
>> Z = X.^2 + Y.^2 - 2*ones(size(X));

>> surf(X,Y,Z);
```

You can also draw a surface "mesh-style":

```
>> mesh(X,Y,Z);
```

Or you can draw a contour map:

```
>> contour(X,Y,Z);
```

One neat thing that MATLAB can do easily is to draw a combined graph and contour map:

```
>> surfc(X,Y,Z);
```

This can also be done in Maple, but it's more difficult.

Another approach: Do everything Maple-style. As you can see, while working with tables of data is very convenient for real-life engineering tasks, it's maybe a little more complicated than what you'd want just to finish your calculus homework. Fortunately, if you own the MATLAB Student Version, you also own parts of the "Symbolic Math Toolbox", which is literally a big chunk of Maple added to MATLAB. (Note that the default professional version of MATLAB does not include the Symbolic Math Toolbox; you need to pay another $200 to get it. Make sure you buy the Student Version!)

If you want to do graphs Maple-style, you first have to declare that the variables you are using are "symbolic", i.e., Maple-style:

```
>> syms x y
```

Once you've declared symbolic variables, any expression you define using those variables becomes a symbolic (Maple-style) function. For example, to define the symbolic one-variable function f and the two-variable function g:

```
>> f = sin(x);

>> g = x^2 + y^2 - 1;
```

You can then plot symbolic functions using the Maple plot commands, prefaced by `ez`, i.e., `ezplot`, `ezsurf`, `ezmesh`, `ezcontour`. Ranges are specified inside (what else) a matrix; `[-2 2]` specifies $-2 \leq x \leq 2$, and `[-2 2 -3 3]` specifies $-2 \leq x \leq 2$, $-3 \leq y \leq 3$.

```
>> ezplot(f,[-2 2]);

>> ezsurf(g,[-2 2 -3 3]);

>> ezsurfc(g,[-2 2 -3 3]);

>> ezmesh(g,[-2 2 -3 3]);

>> ezcontour(g,[-2 2 -3 3]);
```