A SHORT TUTORIAL FOR USING MAXIMA
MARCH 2005

Maxima - a sophisticated computer algebra system
Distributed under the GNU Public License.

Maxima 5.9.1 can be downloaded from the website
http://maxima.sourceforge.net
Steps:
- click on the "download" link in the menu on the left,
- select the appropriate version (i.e. Windows version if you are using Windows),
- click on the "download" link corresponding to the mirror site nearest you.
The download will start; save the Maxima-5.9.1.exe file on your computer and run it; follow the instructions in the wizard.
Note: the default options in the installation wizard will create a shortcut to the program on your desktop.

Start the program.
You should follow this tutorial and execute all the commands on your own as you go along.

The program window is split into two parts. Your commands (input) and output will be in the top window, while the bottom one is a browser that can be used for viewing help files, examples, and even web pages (for this, just type in the url, e.g. http://planetmath.org , into the "url" location and press "Enter".) When the program is first started, the bottom window contains some examples. You may want to read through them.
You can also scroll in both windows using the scroll bars on the right or, as usual, the arrow keys.

Let's familiarize with the syntax of the top (command) window.
First, what does (%i1) mean?
This is the line in which you are to type your command (first input; hence "i1"). After doing so and pressing "Enter", the output will appear on the following line which will begin with (%o1). The next input line is labeled (%i2), etc. (Attention: in this file, the some input lines are skipped, do not worry about this. All you need to worry about is matching the output line number with the corresponding input line number to see what command caused which output.)
For example, typing "2+3;" in the input line and pressing "Enter" produces the following (try it!):

```
(%i1) 2+3;
(%o1)                                5
(%i2)
```

Notice that the command ended with a semicolon! In Maxima, EVERY command must end with a semicolon.
Notice also that the next input line (marked by "(%i2)") showed up automatically. This is where you type your next command.

Now that we are familiar with the Maxima envrionment, let us do something more sophisticated.

For example, we would like to factor the integer 10 into products of powers of primes. Guessing the command "factor" actually produces the result:

```
(%i2) factor(10);
(%o2)                                                          2 5
(%i3)
```

We can also use "factor" to, well, factor polynomials. If the polynomial is irreducible, factoring doesn't change it (note: this also holds for prime numbers, of course. Try it.) Here are some examples:

```
(%i3) factor(x^2+2*x+2);
                                                                2
(%o3)                                                          x  + 2 x + 2
(%i4) factor(x^2+2*x+1);
                                                                        2
(%o4)                                                          (x + 1)
(%i5) factor(200);
                                                                   3  2
(%o5)                                                            2  5
(%i6) factor(x^2+2*a*x+a^2,x^3+1);
                                                                         2
(%o6)                                                          (x + a)
```

If we are unsure how to use the factor command, we can call help by typing:

```
(%i11) describe("factor");
```

pressing "Enter" will then display some options (try it!). Choosing "2" for example, will give us a description of what the command "factor" does for polynomials.
Notice that there is another factorin command, namely "gfactor". Let's try to use it and compare to "factor". Note that we can type two commands in the same input line; the outputs will be displayed one below the other and Maxima will label each one separately.

```
(%i12) factor(x^2+1); gfactor(x^2+1);
                                                                  2
(%o12)                                                           x  + 1
(%i13)
(%o13)                                                    (x - %I) (x + %I)
(%i14)
```

Great: "factor" thinks that x^2+1 is irreducible and gfactor doesn't; so we see that "gfactor" can be used to factor polynomials over the complex numbers.

Now let's learn how to save typing time.
Say we wish to multiply the polynomial that appears in the line (%o12) by another polynomial, say y^2+x*y. Then, instead of retyping the polynomial, we can do the following:

```
(%i14) %o12 * (y^2+x*y);
                                                             2        2
(%o14)                                                      (x  + 1) (y  + x y)
(%i15)
```

But what if we want to see that this output looks like when it is expanded? Simple:

```
(%i15) expand(%o14);
                                                           2  2    2    3
(%o15)                                                    x  y  + y  + x  y + x y
(%i16)
```

All right, now we know all the basic things about polynomials... but what if we wanted to substitute some value for the variables? Then, we have to define the corresponding FUNCTION in terms of the variable, and evaluate it. Here is an example:

```
(%i25) g(x):=3*x^2-2*x+1;
                                                              2
(%o25)                                              g(x) := 3 x  - 2 x + 1
(%i26) g(0);g(1);g(-1);
(%o26)                                                        1
(%i27)
(%o27)                                                        2
(%i28)
(%o28)                                                        6
(%i29)
```

So, the operator " := " is used to assign values or functions.
Next, we need to know how to solve equations. We use the command "solve" for this in one of the two following ways:

```
(%i35) solve(x^2-1);
(%o35)                          [x = - 1, x = 1]
(%i36) solve(x^2+2);
(%o36)                [x = - SQRT(2) %I, x = SQRT(2) %I]

(%i38) solve(x^2+2*x-4,x);
(%o38)               [x = - SQRT(5) - 1, x = SQRT(5) - 1]
```

Note that "sqrt" means "square root". Also, some commands include an "x" at the end of the command, some don't. The "x" that comes after the comma simply means "solve this equation for x". If the variable is obvious, Maxima already knows which variable to solve for. If we try something like "solve(x+y=5);", Maxima will complain, and here is the output:

```
(%i40) solve(x+y=5);
More unknowns than equations -SOLVE
Unknowns given :
[y, x]
Equations given:
[y + x = 5]
 -- an error.  Quitting.  To debug this try DEBUGMODE(TRUE);
(%i41)
```

So, what we need is to tell it which variable to solve for, say "y":

```
(%i41) solve(x+y=5,y);
(%o41)                             [y = 5 - x]
(%i42)
```

That's better. Now, we can also solve a system of equations, such as:

```
(%i43) solve([x+y=5, x-y=1],[x,y]);
(%o43)                           [[x = 3, y = 2]]
(%i44)
```

So, to make a list of equations or variables, use square brackets. In addition, Maxima is smart enough to derive formulae such as this:

```
(%i44) solve(a*x^2+b*x+c,x);
                        2                              2
                  SQRT(b  - 4 a c) + b      SQRT(b  - 4 a c) - b
(%o44)      [x = - --------------------, x = --------------------]
                         2 a                       2 a
(%i45)
```

Of course, since too many letters appeared in my equation, I had to tell Maxima which one is the variable we are solving for.
To combine a few things we have learned so far: we can define a function f, then find its roots using solve. Of course, we can refer to the function by its name

after defining it, or retype the function; the outputs will be the same. Here is an easy example:

```
(%i46) f(x):=x+4;
(%o46)                          f(x) := x + 4
(%i47) solve(x+4,x);
(%o47)                            [x = - 4]
(%i48) solve(f(x),x);
(%o48)                            [x = - 4]
(%i49)
```

Now that we know how to define functions (using "f(x):="), evaluate them at a point (e.g., "f(1);"), and find roots ("solve(f(x),x);"), we might also want to learn how to plot the graphs of functions. Luckily, we can plot in two or three dimensions, and the graphs look quite nice. Maxima needs to know which function we are plotting, and what is the range of the independent variable. Of course we can omit the range of the other variable, or add it if we like. Try, for example, the following commands. (The second one will give an error because range of x is missing!) Note that the graphs of the function will appear in a separate window (called something like "GNU graph"), so we don't see the picture in the output line...

```
(%i49) plot2d(x^3-x^2,[x,-10,10],[y,-5,5]);

(%i50) plot2d(x^3-x^2);

(%i51) plot3d(x^2+y^2,[x,-3,3],[y,-3,3]);
```

Also try clicking on the 3d graphs; you can rotate them to see what the surface looks like from other perspectives.
We can also add a grid to our picture:

```
(%i52) plot3d(x^2+y^2,[x,-3,3],[y,-3,3],[grid,1,1]);
```

Try changing the numbers 1,1 in the grid to see what they represent. Note that the commands above could've been shorter if we first defined the function and then used its name to plot it; so let's try that now:
1. define a function f(x,y):=3-x+y for example (or choose another one you like),
2. use the plot3d command and instead of the equation, write the name "f(x,y)" as the first entry in parenthesis,
3. choose a grid if you like and any range of values you like.
If you don't like the graph, try changing ranges.

We will now briefly cover summation in Maxima. We would like to know how to compute finite and infinite sums, get formulae and values for them.
For example, we can ask Maxima to sum first 10 integers:

```
(%i13) sum(x,x,1,10);
(%o13)                               55
```

The first entry "x" is the expression we are summing, the second is our counter, and the last two are the range (i.e. add "x" for values 1 to 10). We can also ask the following:

```
(%i14) sum(x,x,1,n);
                                     n
                                   ====
                                   \
(%o14)                              >    x
                                   /
                                   ====
                                   x = 1
```

- but this doesn't give us a formula for how to compute this sum for an arbitrary integer n. To see the formula, we put "ev(* ,simpsum);" where * represents the command for the sum given above.

```
(%i15) ev(sum(x,x,1,n),simpsum);
                                    2
                                   n  + n
(%o15)                             ------
                                     2
(%i16)
```

Here are more examples of finite and infinite summation. Note that infinity is denoted by "inf", and if the infinite sum cannot be calculated, Maxima outputs "undefined".

```
(%i1) sum(2^x,x,1,n);
                                    n
                                   ====
                                   \      x
(%o1)                               >    2
                                   /
                                   ====
                                   x = 1
(%i2) sum(2^x,x,1,10);
(%o2)                                2046
(%i3) ev(sum(2^x,x,1,n),simpsum);
                                   n + 1
(%o3)                             2      - 2


(%i7) ev(sum((1/2)^x,x,1,inf),simpsum);
(%o7)                                1
(%i8) sum((1/2)^x,x,1,inf);
                                   INF
                                   ====
                                   \     1
(%o8)                               >    --
                                   /      x
                                   ====  2
                                   x = 1
(%i9) sum(2^x,x,1,inf); ev(sum(2^x,x,1,inf),simpsum);
                                   INF
                                   ====
                                   \      x
(%o9)                               >    2
                                   /
                                   ====
                                   x = 1
(%i10)
(%o10)                                INF
(%i11) sum((-1)^x,x,1,inf); ev(sum((-1)^x,x,1,inf),simpsum);
                                 INF
                                 ====
                                 \          x
(%o11)                            >    (- 1)
                                 /
                                 ====
                                 x = 1
(%i12)
(%o12)                             UNDEFINED
```

We have come to an end of this Maxima introductory tutorial. Hopefully now you will be able to explore other commands that Maxima has to offer; remember, there is extensive help available and plenty of examples are also included with the program.

But as you use the program, you might wish to save the results for use later or even to make handouts in class. We have found that the easiest and least painful way to do this (and we have created this tutorial the same way) is the following:
- execute the desired commands in Maxima
- open Notepad (Start-> All programs -> Accessories -> Notepad)
- highlight the commands and output that you'd like to copy and save
- copy this using either Ctrl+C or select Edit->Copy in the top Maxima window
- click inside the Notepad window and paste the contents (either Ctrl+v or using Edit->Paste in the Notepad window)

Add any comments you like to the file.
You are now ready to save your file; it will be saved as text (with extension .txt) and is best viewed in Notepad. You may attempt to use other programs such as Microsoft Word, but we have had difficulties with text alignment and most output doesn't look the same. Stick to Notepad; you can even print from it.

Good luck and have fun with Maxima!!

APPENDIX 1:

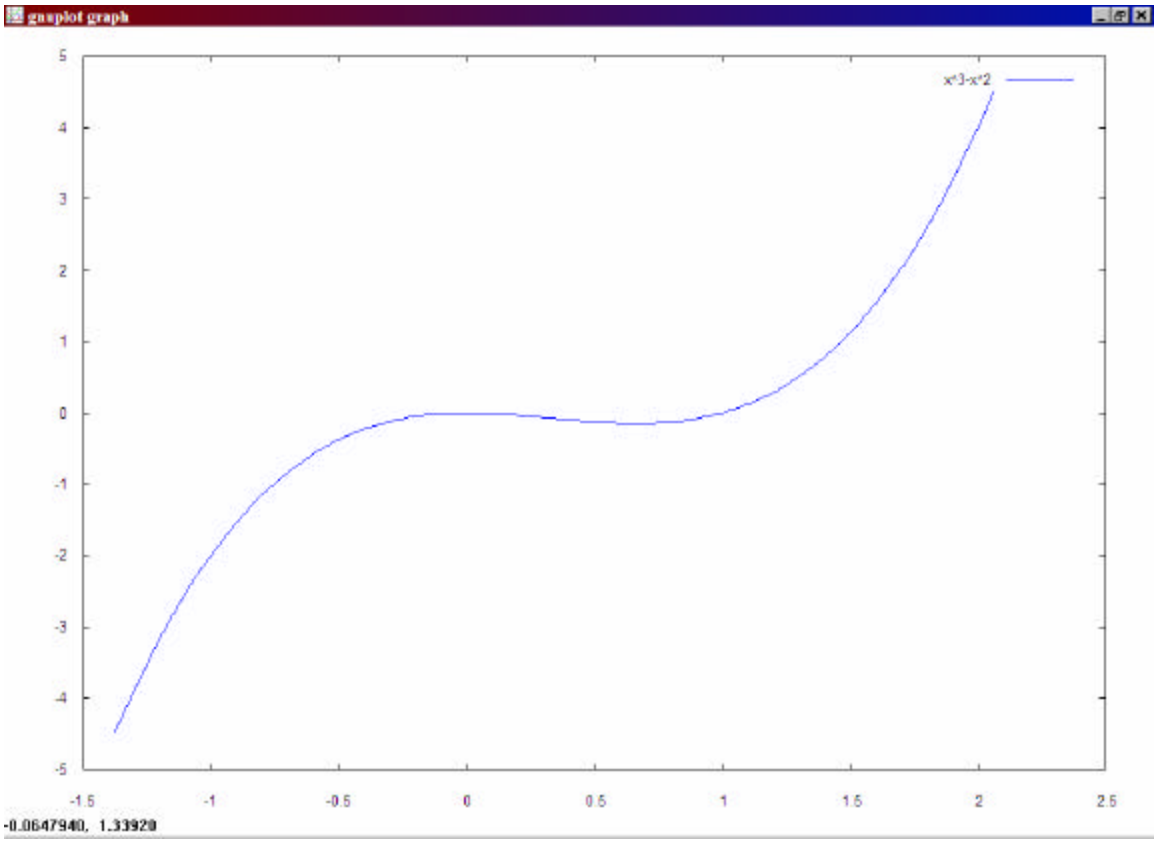
A list of more cute commands you might be interested in:

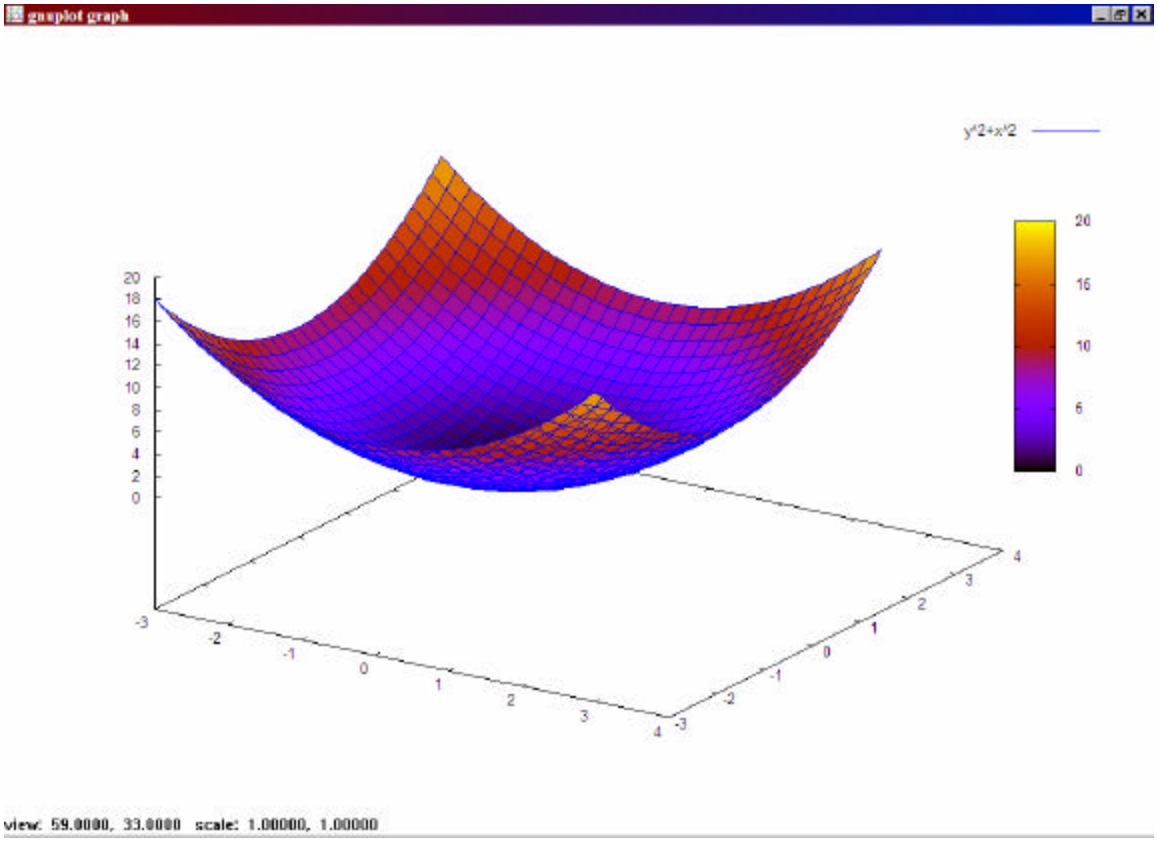
```
(%i62) gcd(4,36);
(%i63) f(x):=(x+3)^2*(x^3-7)^2; g(x):=(x+3)*x*(x+2); gcd(f(x),g(x));
```

(more to be added.)

APPENDIX 2:

We include the two graphs that were generated in this tutorial; one two- and one three-dimensional.


gnuplot graph
x^3-x^2
5
4
3
2
1
0
-1
-2
-3
-4
-5
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
-0.0647940, 1.33920


gnuplot graph
y^2+x^2
20
18
16
14
12
10
8
6
4
2
0
-3
-2
-1
0
1
2
3
4
20
16
10
6
0
view: 59.0000, 33.0000 scale: 1.00000, 1.00000