Appendix: Quick reference to Maxima

1 Getting started

If you move your cursor to right here, you will see that this command is written in an entry (bar on the left) that is Text (toggle up top says "Text"). If you move your cursor to "Getting started", right above this though, you see that is a section.

If you move the cursor right below this sentence, you will see that it gives you a horizontal line on the left, that's a place where you can write input. Type "2+2;" but without the " " (we'll use " " to talk about what we want you to enter into maxima) and then hit enter (or shift return):

Maxima is very handy because it can handle numbers (like 2+2) but also equations. To see what we mean, move your cursor into the next line and hit enter (or shift return):

```
(%i2)   x+x;
(%o2)   2 x
(%i1)   2+2;
(%o1)   4
```

2 Getting help

"describe(command)", or "? command" (with a space after the question mark): Prints to the screen the documentation of a particular command, e.g. "describe(integrate)" prints the documentation of the integrate() function.

"describe(string, inexact)" or "?? inexact" (with a space after the question marks): Prints to the screen a list of items documented in the manual which contain "string" as part of their name.

"apropos("string")" returns a list of Maxima commands that have "string" appearing anywhere in them. For example,

"apropos("exp")" retruns a list of functions including expand(), exp(), and ratexpand().

3 Basic commands

Operations
"*" Times command (2*3; gives 6).
"^" Power command (2^3; gives 8).
"!" Factorial (3!; gives 6).

Parentheses
"[ ]" Denotes a list, e.g. [2,3,4]. Can also denote the argument of a recurrence equation, e.g. n[3].
"( )" Has two functions. It can place variables together, e.g. (1+x)/(1-x) takes 1+x over 1-x. It is also used to denote arguments of functions, e.g. exp(3) is %e^3.

Grabbing output
"%o#" Grabs the previous output number #
"%" Grabs the previous output regardless of number. It is also used for some built-in symbols, e.g. %pi, %e, etc.

Other basic commands
":" Defines a variable. For example: r : 10*x assigns the value 10*x to the symbol (variable) r. You can then perform operations on r, e.g. 2*r returns 20*x. See also: section "Writing equations in Maxima".
"$" Omits the output of a particular operation. For example, a : 3*2; assigns the value 6 to the symbol a and returns the value 6, whereas 3*2$ doesn't returns anything (but still performs the assignment).
"numer" Used to return numerical (instead of symbolic) values, e.g. 100*%pi, numer; returns 314.15926...

Substitutions
"f, object1 = object 2" Tells maxima to substitute object1 with object2 in the function, e.g. 3*x^2, x=2*y+z; gives 3*(z+2y)^2
"subst(object1 = object2, f)" Same as the above

4 Avoiding conflict with Maxima

Maxima tends to use lowercase letters for functions, so you can use initial upper case names for your functions and variables.

If you refer to previous outputs using %, it can be difficult to know exactly what your previous entry was. It is safer to assign a name to the output and then refer to this name later.

For example:

```
➔ MyDerivative : diff(a·sin(b·x), x);
  wxplot2d(subst([a = 1, b = 3], MyDerivative), [x, 0, %pi]);
```

5 Functions and constants in Maxima (a small fraction!)

Absolute value, Logarithm, Square Root, Trigonometry
"abs(x)" Takes the absolute value of x
"log(x)" Takes the natural log of x. Maxima doesn't have a built-in function for base 10 logarithm or other bases. You can calculate the base-n logarithm of x as log(x)/log(n);
"sin(x)", "cos(x)", "tan(x)" Trigonometric functions
"arcsin(x)", "arccos(x)", "arctan(x)" Inverse trigonometric functions
"sqrt(x)" Square root

In-built symbols:
"%e" The exponential constant, 2.71838... %e^x can also be invoked using exp(x).
"%i" The square root of negative 1
"%pi" 3.14159...
"inf" Infinity

6 Writing equations in Maxima

"x : y" Sets x to y immediately and from then on. Use kill(x) to unassign x, e.g. plot1 : wxplot2d(x^2, [x,0,10])
"'x : y" Does nothing until x is called, at which point x is assigned the value y
"x = y" Defines an equation
"f(x) := y" Defines a function f, e.g. f(x) := x^2
"f(x)" This gives the function evaluated at x, e.g. f(3), numer; gives 9 in the above example

7 A list of helpful commands

Memory
"kill(symbol1)" Clears variable or function definitions
"kill(all)" Clears all variable or function definitions from memory

Algebra
"expand(expr)" - expands an expression, e.g. expand((1+x)^2) returns x^2 + 2*x + 1
"factor(polynomial)" - Self explanatory. E.g. factor(x^2+2*x+1) returns (x+1)^2
"collectterms(eqn, term)" - Collects parts of an equation involving a term and factors them separately. E.g. collectterms(a-b+a*x-2*b*x+a^2*x^2, x); returns a^2*x^2+(a-2*b)*x-b+a
""logcontract(expr) - contracts a sum of logs into a single log expression
"ratsimp(expr)" Simplifies a rational expression, e.g. a polynomial. E.g. ratsimp((x+2)*(x-2)); returns x^2-4. You can continue simplifying the expression until no further change occurs by calling fullratsimp() instead.
"radcan(expr)" - meaning "radical cancelation", it is useful in simplifying expressions containing logs, exponentials and radicals. E.g. radcan((exp(x)-1)/(exp(x/2)+1)); simplifies an expression that ratsimp() cannot simplify.
"sum(f, i, imin, imax)" - Sums a function f which depends on i, from imin to imax. E.g. sum(i+1, i, 1, 4) returns 14

Calculus
"diff(f,x)" - Takes the partial derivative of f with respect to x. E.g. diff(x^2+y*log(x), x) returns 2*x + 1/2
"diff(f,x,n)" - Takes the nth derivative of f with respect to x. E.g. diff(x^2+log(x), x, 2) returns 2 - 1/(x^2)
"integrate(f,x)" - Finds the indefinite integral of f with respect to x. E.g. integrate(log(x),x) returns x*log(x)-x
"intgrate(f,x,xmin,xmax)" - Finds the definite integral from xmin to xmax. E.g. integrate(log(x),x,1,6) returns 6*log(6)-5

Solving a recursion equation for y as a function of x (in this example, y(t+1) = x*y(t)), SYMBOLICALLY, with initial value y(0) = y0:
```
load("solve_rec")$
solve_rec(y[t+1] = x*y[t], y[t], y[0]=y0);
```

Solving a differential equation for y as a function of x (in this example, y'(t) = x*y(t)), SYMBOLICALLY, with initial value y(0) = y0:
```
eqn: 'diff(y(t), t) = x*y(t);
atvalue(y(t), t = 0, y0);
desolve(eqn, y(t))
```

Solving a differential equation NUMERICALLY: rk uses the Runge-Kutta method
```
rk(t-x^2,x,1,[t,0,8,0.1])$
```

Solving an equation symbolically:
"solve(eqn, vars)", e.g. solve(x^2 = 3, x)

Can also solve a system of equations, e.g. solve([x^2=3, y+3=x], [x, y])

Solving an equation numerically within a given interval. For example:
"find_root(x^2 = 7, x, 0, 100);" returns 2.6457...

Plotting:
"wxplot2d(f, [x,xmin,xmax])" - plots f versus x on the interval [xmin,xmax]
"wxplot2d([discrete, list1, list2], [style, points])" - plots points with x-coordinate given by list1, and y-coordinate given by list2
"wxdraw2d" - is a more powerful function that can draw multiple functions and implicit functions (like a circle)

Others