Maxima Commands

Multivariable Calculus, Quest University

Learning Maxima isn't all that hard if you can access the commands you need, relatively easily. This document is being compiled by a series of multivariable calculus classes as we go, to make available in a simple format the syntax for the commands you need. I suggest you keep this document around, maybe even print it, so that you don't have to worry too much about writing down obscure commands in class. Everything in bold-face italics is, obviously, to be filled in by you.

If you discover a new command, or a better way to do anything listed below, let me know and I'll add it to the list.

2D parametric plots:
plot2d([parametric,***x-function***,***y-function***,[t,***t-min***,***t-max***]])

Multiple 2D parametric plots:
plot2d([[parametric,***x-function1***,***y-function1***,[t,***t-min***,***t-max***]],[parametric,***x-function2***,***y-function2***,[s,***s-min***,***s-max***]]])

Adding more tick marks to a graph:
plot2d([parametric,***x-function***,***y-function***,[t,***t-min***,***t-max***]],[nticks,200])

Plotting one or more implicitly defined expressions (in terms of *x* and *y*):
load(implicit_plot)$
implicit_plot(expression,[x,***x-min***,***x-max***],[y,***y-min***,***y-max***])
implicit_plot([expression1,expression2,…],[x,***x-min***,***x-max***],[y,***y-min***,***y-max***])

3D parametric plots:
load(draw)
draw3d(nticks=200,parametric(***x-fnctn***,***y-fnctn***,***z-fnctn***,t,***t-min***,***t-max***))

Solving systems of equations (in this case 3 eqns, 2 variables):
solve([***eqn1***,***eqn2***,***eqn3***],[***var1***,***var2***])

Making a polar plot:
load(draw)
draw2d(nticks=200,xrange=[***x-min***,***x-max***],yrange=[***y-min***,***y-max***],polar(***r-fnctn***,t,***t-min***,***t-max***))

Making two polar plots on one graph:
draw2d(nticks=200,xrange=[***x-min***,***x-max***],yrange=[***y-min***,***y-max***],polar(***first r-fnctn***,t,***t-min***,***t-max***),polar(***second r-fnctn***,t,***t-min***,***t-max***))

Taking limits:
limit(***fnctn***,***variable***,***limit value***)

Simplifying a complicated ratio:
ratsimp(***expression***)

Simplifying a trigonometric expression:
trigsimp(***expression***)
trigreduce(***expression***)

Dot product of two vectors:
[***vector1***].[***vector2***]

Cross product of two vectors:
load(vect)
express([***vector1***]~[***vector2***])

Referring to components of vectors:
LengthOfFred:sqrt(Fred[1]^2+Fred[2]^2+Fred[3]^2)

Approximating a definite integral:
quad_qags(***function***,x,***x-bound1***,***x-bound2***)
The first component of the result is the answer. The other components give information about the expected accuracy of the result.

3D surface plot:
plot3d(***function***,[x,***x-min***,***x-max***],[y,***y-min***,***y-max***])

More grid points on a 3D surface plot:
plot3d(***function***,[x,***x-min***,***x-max***],[y,***y-min***,***y-max***],[grid,50,50])

Contour plot:
contour_plot(***function***,[x,***x-min***,***x-max***],[y,***y-min***,***y-max***])

Tell Maxima to use more level curves in its contour plots:
set_plot_option ([gnuplot_preamble,"set cntrparam levels 12"])$
Then execute the contour_plot command. The $ at the end suppresses Maxima's output, which you don't need to see here unless you really want to.

Multiple surface plots:
load(draw)
draw3d(color=red,explicit(***fnctn1***,x,***x-min***,***x-max***,y,***y-min***,***y-max***),color=blue,explicit(***fnctn2***,x,***x-min***,***x-max***,y,***y-min***,***y-max***)

Plotting a surface in spherical coordinates:
plot3d(***rho function***,[theta,***theta-min***,***theta-max***],[phi,***phi-min***,***phi-max***],[transform_xy,spherical_to_xyz]);

Dealing with square roots within bounds of integrals:
assume(n>0)
integrate(***function***,x,0,sqrt(n))

To declare a named variable to be a constant:
declare(***x***,constant)

Approximating a double integral:
The following command evaluates an integral for x between x-min and x-max, and y between r(x) and s(x). All symbols are necessary!
quad_qags('quad_qags(***fnctn***,y,***r(x)***,***s(x)***)[1],x,***x-min***,***x-max***)

Plotting a parametrically defined surface:
plot3d([***x-fnctn***,***y-fnctn***,***z-fnctn***],[u,***u-min***,***u-max***],[v,***v-min***,***v-max***])

Plotting multiple parametrically defined surfaces:

load(draw)

draw3d(color=red, parametric_surface(***fnctn1_x***,***fnctn1_y***,***fnctn1_z***,u,***u-min***,***u-max***,v,***v-min***,***v-max***), color=blue,

parametric_surface(***fnctn2_x***,***fnctn2_y***,***fnctn2_z***,u,***u-min***,***u-max***,v,***v-min***,***v-max***))