Maple 9.5 Interface Notes

- Each command in Maple must end with a semicolon (;) or a colon (:). Using the colon suppresses output. It's usually a good idea to use the colon when loading a package (e.g. with(plots):), or when saving a plot to a variable.
- To enter multiple commands while suppressing evaluation, use **<shift>-<Enter>** between commands. When you're ready to evaluate, press **<Enter>**.
- To delete a line, try highlighting the line, then pressing **<Delete>**. If that doesn't work, try **<Ctrl>-<Delete>**.
- If you want your input to be formatted in standard math notation click on  while the cursor is on the input line. In this mode you might find the Expression Palette useful (click on **View-Palettes-Expression Palette**).
- The **%** symbol refers to the last result which Maple returned. **%%** refers to the result before the last. **%%%** goes back 3 results. Et cetera!
- The constant π (the ratio of a circle's circumference divided by its diameter) can be entered by typing **Pi** (but **not** pi !).
- The imaginary unit ***i***, is entered in upper case: **I**
- The base *e* of the natural exponential function, e^x, is easiest to define by entering e:=exp(1); (exp(x) returns e^x).
- The = sign is used for defining equations. It is not used to assign expressions to variables. So **solve**(*x^2+5*x+6=0,x*); uses the = sign in this way.
- To assign expressions to variables, use the := operator. For example, *y:=x^2+5*x+6;* stores *x^2+5*x+6* in the variable *y*.
- After assigning a value to a variable, the variable becomes a constant, and can no longer be used as an independent variable in plotting, differentiating, or integrating. To delete a value stored in a variable, *x*, use **unassign**('*x*'); Note that *x* is surrounded by 'apostrophes'.
- To create a function, there are two approaches: the first uses the syntax:

  dependent_variable:=independent_variable -> expression; Thus, to create $f(x) = 2x - 3$, you would enter **f:=x -> 2*x – 3;**
- The % operator (last result) cannot be used to define a function. If you need to use % to define a function, use the unapply command (see below).
- All functions and variables in Maple are Case Sensitive. So X is different from x, and it is possible to create functions like T(t).
- When computing odd indexed roots of negative numbers, Maple returns the principal root, which is imaginary. To get the real nth root of x, use **surd**(x,n). So, if you want to plot $y = \sqrt[3]{x}$, use **plot**(**surd**(x,3), x=-8..8); By the way, the archaic word *surd* means *irrational number*, or particularly, *radical*.

Useful Maple Commands

evalf(numerical-expression); approximates *numerical-expression*.
  evalf(20!); returns .2432902008 * 10^19
eval(numerical-expression); evaluates a *numerical-expression* exactly.
  eval(20!); returns 2432902008176640000
eval(*expression, x=value*); substitutes *value* for *x* in *expression*.
eval(*expression*, {*x=value1, y=value2*, etc. ...}); performs multiple substitutions.
  eval(x^2-y, {x=7, y=2}); returns 47.
subs(*x=a,expression*) substitutes each *x* with an *a* in expression.
  subs(sin(x)=y, sin(x)/(1-sin(x))^(1/2)); returns y/(1-y)^(1/2).
assume(*variable,domain1*); restricts *variable* to values in *domain*.
additionally(*variable,domain2*); puts further restrictions on *variable*.
  assume(*t*, real); forces *t* to be a real number.
  additionally(t,positive); forces to be positive in addition to being real.

Using Maple for Common Mathematical Operations

x*y;	* is used for multiplication.
x/y;	/ is used for division.
x^y;	returns x^y
sqrt(x);	returns $\sqrt{x}$.
ln(x);	returns $\ln(x)$.
log[b](x);	returns $\log_b(x)$.
exp(x);	returns e^x
surd(x,n)	returns $\sqrt[n]{x}$ (real roots only)
sin(x); cos(x); tan(x);	trigonometric functions
arcsin(x); arccos(x); arctan(x);	inverse trig functions

Maple assumes symbolic variables to be complex unless otherwise specified. Certain operations (like the dot product) yield unexpected results if we forget to tell Maple that our variables are real! Also, many operations are not valid over the set of all real numbers. So **expand**(ln(*a*b*)); doesn't expand unless we execute **assume**(*a, positive*); and **assume**(*b, positive*); first. Finally, whenever Maple outputs an expression in terms of a variable which has associated assumptions, these variables will be displayed with a trailing tilde (so if *t* is assumed positive, Maple will display *t~*). To turn this feature off, click **File-Preferences** then click the **I/O Display** tab, and under **Assumed Variables**, click on **No Annotation**. Click **Apply to Session** to save this setting.

expand(*expression*); distributes *expression* completely.
  expand((2*x-3)*(x-4)); returns 2*x^2-11*x+12.
normal(*expression*); collects fractions, simplifies complex fractions, and reduces fractions.
  normal(2/x+x/2); returns (4+x^2)/(2*x).
simplify(*expression*); uses many rules to find the "simplest" form for *expression*.
  simplify(sqrt(5*sin(2*x)^2+5*cos(2*x)^2)); returns sqrt(5);
collect(*expression,variable*); combines like terms with respect to *variable*.
  Collect(a*x+2*a^2*x+4*x^2,x); returns (a+2*a^2)*x + 4*x^2.
factor(*expression*); attempts to factor *expression* over the rationals.
factor(*expression,field*); returns an approximate factorization over *field*.
  factor(2*x^3-x^2-5*x+3); returns (x^2+x-1)*(2*x-3).
  factor(2*x^3-x^2-5*x+3,real); returns 2.*(x+1.618033989)*(x-.6180339887)*(x-1.500000000).
  factor(x^3+1,complex); returns (x+1.)*(x-.5000000000+.8660254038*I)*(x-.5000000000-.8660254038*I)
ifactor(*integer*); factors *integer* into a product of primes.
  2^101-1; returns 25353012004564588029934064107 51, and then, **ifactor**(%); returns (7432339208719)*(341117531003194129).
convert(*expression, form*); converts *expression* into an equivalent expression of type *form*.
  The most useful form is *parfrac* for partial fractions, e.g. **convert**(1/(x^2-4), *parfrac, x*);
f:=**unapply**(expression,variables); creates *f*, a function of *variables*. The main reason for using unapply over the previous method for creating functions is that the % operator may be used here.
  f:=unapply(x^2-4,x); g:=unapply(x^2+y^2,x,y); h:=unapply(%,x);

solve(*equation,variable*); solves for *variable* in *equation*.

When a solution of an equation is too messy or not exact, Maple will use the RootOf function to represent each of the solutions. For example, solve(x^4+x^3+x^2+2*x+7,x); returns RootOf(_Z^4+_Z^3+_Z^2+2*_Z+7) which is the set of all roots of the equation. To see the values use the function **allvalues(%);** to see all of the roots of the equation.

solve({*equation list*},{*variable list*}); solves a system of equations.

solve(inequality,variable); solves for *variable* in *inequality*.

solve(abs(x-4)>3,x); returns RealRange(Open(7),infinity), RealRange(-infinity,Open(1)), which is Maple for $(-\infty, -7) \cup (7, \infty)$.

solve({*inequality list*},{*variable list*}); solves a system of inequalities.

diff(f(x), x); computes $\frac{d}{dx} f(x)$, **diff**(f(x,y), x, x); computes $\frac{d^2}{dx^2} f(x)$, **diff**(f(x, y), x, y); computes $\frac{\partial^2}{\partial y \partial x} f(x, y)$.

int(f(x),x); computes $\int f(x)dx$, while **int**(f(x),x = a..b); computes $\int_a^b f(x)dx$.

dotprod(*v, w*); computes the dot product of vectors *v* and *w*. The linear algebra package must be loaded first using: **with**(*linalg*):

norm(*v, n*); computes the *n*th norm (a.k.a magnitude) of vector *v*: $norm(v, n) = \sqrt[n]{\sum v_i^n}$. If n is omitted the infinite norm is used (where $n \to \infty$). Of course, we need $n = 2$ for the length of a vector in $\Re^2$ or $\Re^3$. The linear algebra package must be loaded for this operation, do this by executing **with**(*linalg*):

limit(*f(x),x=a*); computes the limit of *f(x)* as *x* approaches *a*.

sum(*f(i),i=m..n*); computes the sum on *f(i)* as *i* ranges from *m* to *n*.

Two Dimensional Plots

plot(*function, x=xmin..xmax, options*); plots *y* = function with independent variable x ranging from *xmin* to *xmax*.

plot([*x(t),y(t),t=tmin..tmax*], *options*); plots a graph represented parametrically, with parameter t ranging from *tmin* to *tmax*.

plot({list of functions to plot},*x=xmin..xmax,options*);

implicitplot(*f(x,y)=0,x=xmin..xmax,y=ymin..ymax,options*); plots equations where y is defined implicitly. Execute **with**(plots): first!

Allowable options include: (default options are printed in **boldface**).

axes = *frame*, *boxed*, ***normal***, or *none*	Sets the type of axes to display.
color = *blue*, *black*, *red*, etc.	
discont = *true* or ***false***	Tells Maple to look for discontinuities on the graph.
labels = [x, y]	Sets the names to display for the vertical & horizontal axes.
scaling = *constrained* or ***unconstrained***	Constrained scaling creates a 1:1 aspect ratio (circles look like circles – not ellipses).
title = "Title for your graph"	
view = [*xmin..xmax, ymin..ymax*]	Sets the coordinates of the corners of the view "window".
coords = *polar*, ***cartesian***	Sets the coordinate system.
y = *ymin..ymax*	Sets the max & min y-values for view "window".

Examples:

plot(sin(x), x=-2*Pi..2*Pi, color = black, title="y = sin(x)");

plot(1/x, x=-1..1, y=-2..2, discont = true);

plot([2*cos(t), 2*sin(t), t=-2*Pi..2*Pi], view = [-3,3,-3,3], scaling = constrained);

plot({sin(x), 1/x, [2*cos(t), 2*sin(t), t=-2*Pi..2*Pi]}, x=-2*Pi..2*Pi, y=-3..3, scaling = constrained, color = black);

inequal(linear_inequality, *x=xmin..xmax, y = ymin..ymax);* Note: **with**(*plots*): must be executed first!

inequal({set of linear_inequalities}, *x=xmin..xmax, y = ymin..ymax);* Note: **with**(*plots*): must be executed first!

inequal(x+2*y>1, x=-4..9,y=-5..4);

inequal({x+2*y>1,x-y<6},x=-4..9,y=-5..4);

Three Dimensional Plots

plot3d(f*(x,y), x = xmin..xmax, y = ymin..ymax, options*); plots z = f(*x, y*).

plot3d([x(u,v), y(u,v), z(u,v)] plots surface defined parametrically.

plot3d({list of functions of x & y}, *x = xmin..xmax, y = ymin..ymax, options*); plots z = f(*x, y*).

If *xmin* and *xmax* are constants, then *ymin* and *ymax* may be functions of *x*. If *ymin* & *ymax* are constants, then *xmin* & *xmax* may be functions of *y*.

implicitplot3d(*f(x,y,z)=0, x = xmin..xmax, y = ymin..ymax, z = zmin..zmax, options*); where z is defined implicitly. Execute **with**(plots): first!

Allowable options for **plot3d** listed below: (default options are printed in **boldface**).

axes = *boxed, normal, frame,* ***none.***	*grid* = [*m, n*]
color = *red, blue, black,* etc.	*labels* = [*x, y, z*]
style = *contour, point, hidden,* ***patch,*** *wireframe,* *patchnogrid, patchcontour, line.*	*scaling* = *constrained* or ***unconstrained***
	title = "title for plot"
contours = number of contours on a contour plot.	*view* = [*xmin..xmax, ymin..ymax, zmin..zmax*] (or *view* = *zmin..zmax*)
coords = ***cartesian****, cylindrical, spherical.*	

Examples: **plot3d**(4-(x^2+y^2), x=-2..2, y=-sqrt(4-x^2)..sqrt(4-x^2), scaling = constrained, grid = [20,20]);

plot3d([sin(phi)*cos(theta),sin(phi)*sin(theta),cos(phi)], phi = 0..Pi, theta = 0..2*Pi, scaling = constrained);

To combine plots, store the plot in a variable, then use the display or display3d command to view them together. The **display** and **display3d** commands are in the *plots* package, so execute **with**(*plots*): first.

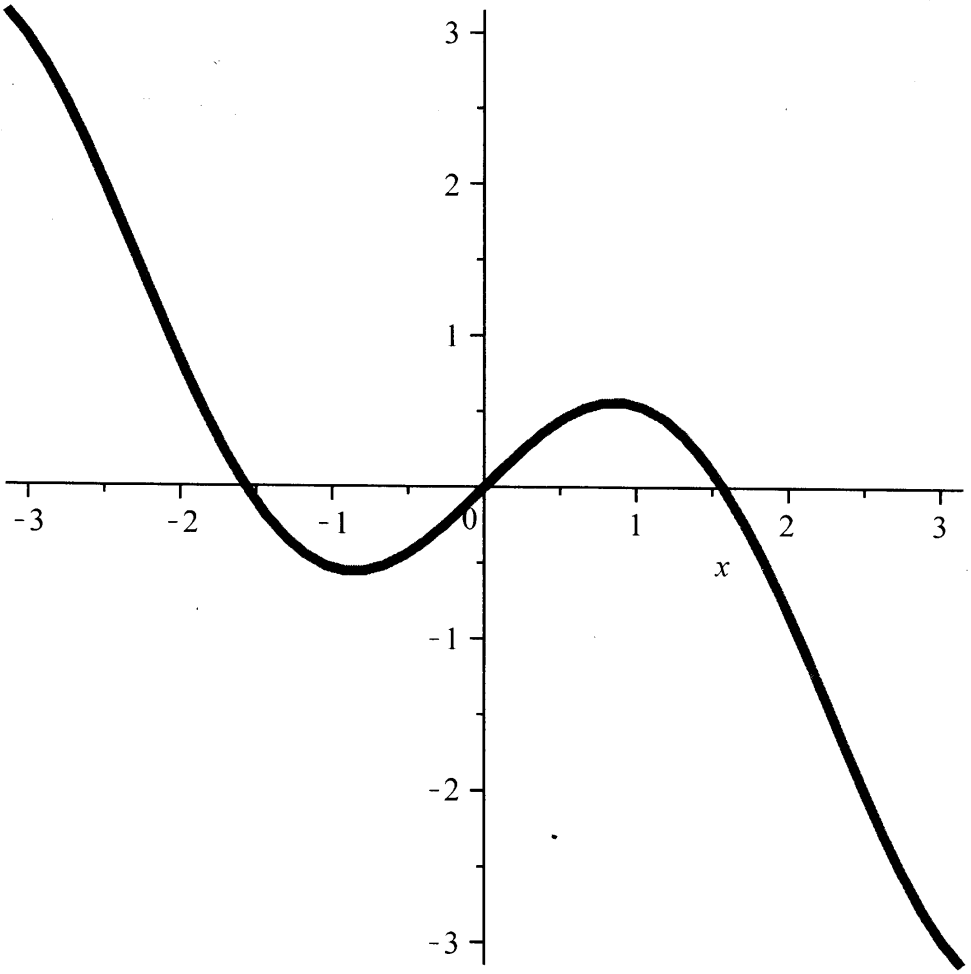
Example: **with**(*plots*): p1:=**plot3d**({sqrt(1-x^2), -sqrt(1-x^2)}, x = -1 .. 1, y = -x .. x):

p2:=**plot3d**({sqrt(1-y^2), -sqrt(1-y^2)}, x = -y .. y, y = -1 .. 1): **display3d**(p1, p2);

Example $f(x) = x\cos x$, $x \in [-\pi, \pi]$

graph $y = f(x)$

```
> restart : with(plots) : with(plottools) :
> a := x·cos(x) :
> b := plot(a, x = -Pi..Pi, color = blue, thickness = 4) :
> display(b);
```



find f'

```
> d := diff(a, x);
```

$$d := \cos(x) - x\sin(x) \quad (1)$$

solve $f'(x) = 0$

```
> eqn := d = 0;
```

$$eqn := \cos(x) - x\sin(x) = 0 \quad (2)$$

```
> soln1 := fsolve(eqn, x, -2..0) :
> soln2 := fsolve(eqn, x, 0..2) :
> soln1, soln2;
```

$$-0.8603335890,\ 0.8603335890 \quad (3)$$

evaluate f at the critical points

```
> eval(a, x = soln1);
```

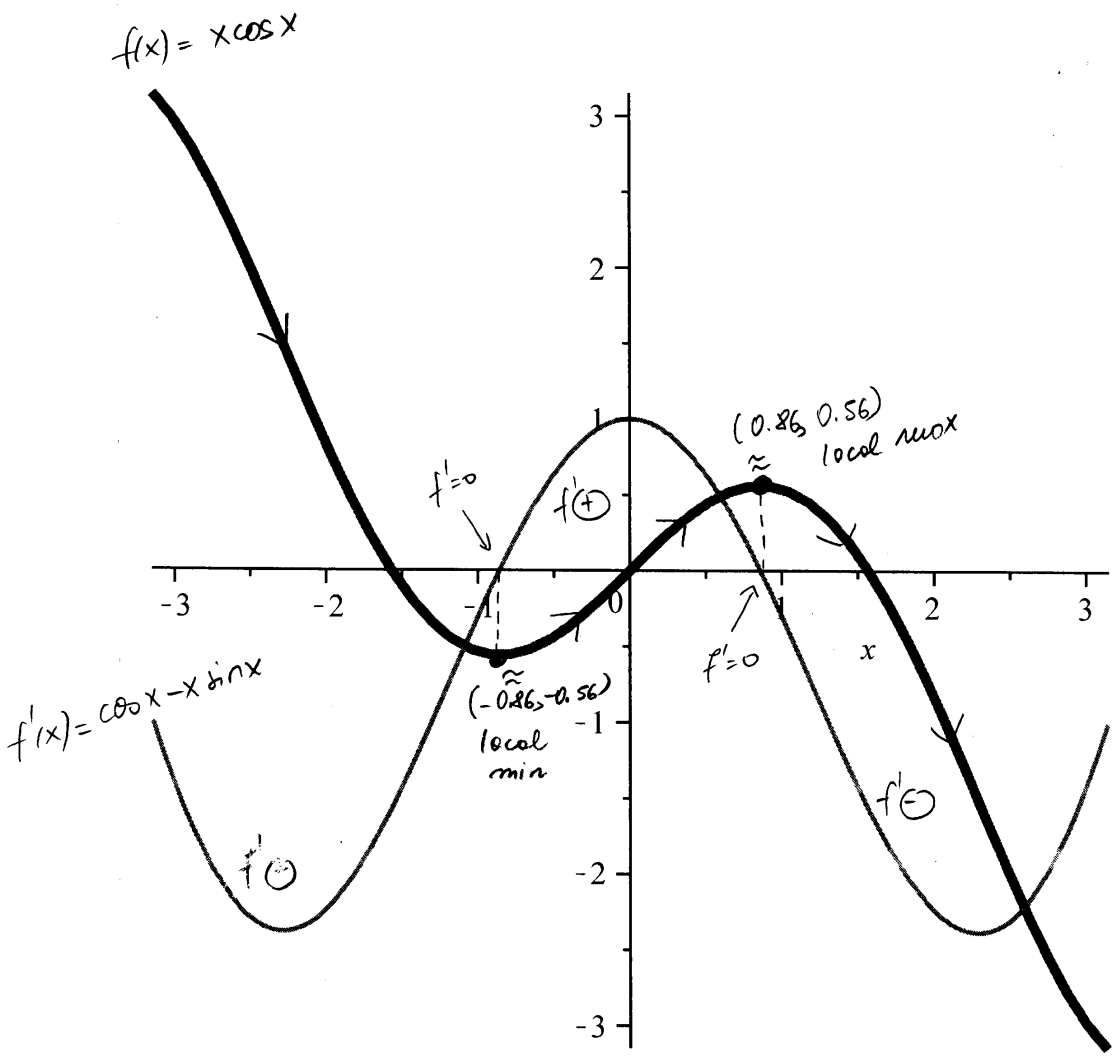
$$-0.5610963382 \quad (4)$$

```
> eval(a, x = soln2);
```

$$0.5610963382 \quad (5)$$

graph f, f'

```
> g := plot(d, x = -Pi..Pi, color = red, thickness = 2) :
> display(b, g);
```



Answers :

b) Domain: $x \in \mathbb{R}$

c) Critical numbers: $x \approx -0.86$ and $x \approx 0.86$

d) - local minimum value is $f(-0.86) = -0.56$;
- local maximum value is $f(0.86) = 0.56$;
- the function is decreasing on $[-\pi, -0.86] \cup [0.86, \pi]$ and increasing on $[-0.86, 0.86]$