



School of Engineering
Brown University

Dynamics and Vibrations

Mupad tutorial

ENGN40 will be using Matlab 'Live Scripts' instead of Mupad.

You can find information about 'Live Scripts' in the ENGN40 [MATLAB tutorial](#)

1. What is Mupad?

Mupad is a GUI driven MATLAB package that helps you do algebra, calculus, as well as to graph and visualize functions. As you know, MATLAB is good for writing simple programs and working with numbers, but is cumbersome for doing symbolic calculations. In contrast, Mupad works with symbols by default, and has a nice menu-driven interface.

2. Starting Mupad

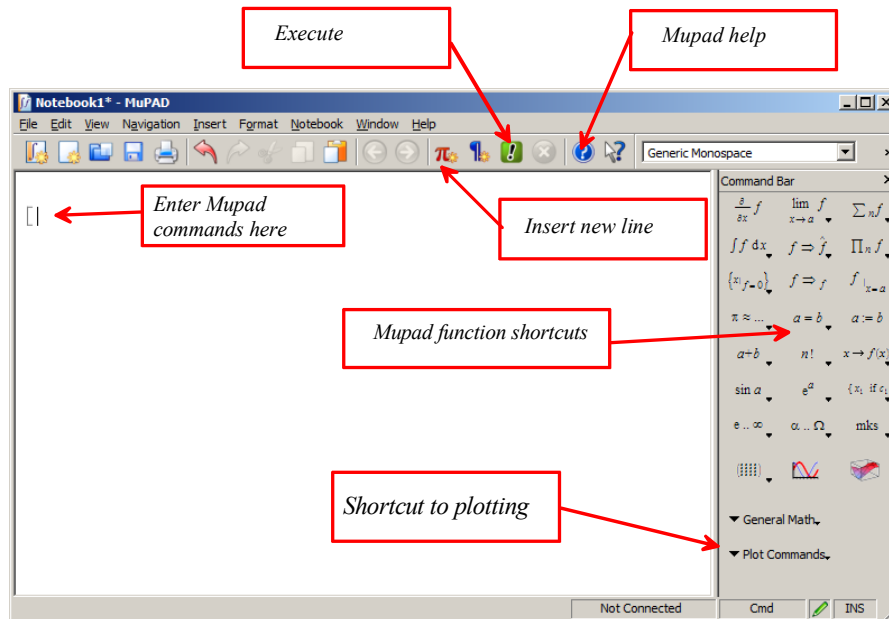
You run Mupad by first starting MATLAB (see the Matlab tutorial if you don't know how to do this), then typing

```
>> mupad
```

in the MATLAB command window. Start mupad now, and type in the commands as they appear in the tutorial. You might find it helpful to experiment and explore the various functions on your own as well.

3. Basic Mupad window

You should see the GUI shown below. Most of the buttons should be self-explanatory. Try clicking a few at random to see what happens...



4. Simple arithmetic calculations

Just like MATLAB itself, Mupad can be used as a calculator. Try the following commands

```
[ 2+2
  4
  sin(PI/3)
   $\frac{\sqrt{3}}{2}$ 
  30!
  26525285981219105863630848000000
  sin(PI)
  0
  gamma(0.1)
  9.513507699
  besselJ(0.1,1)
  0.770765187
```

Here, the gamma function and besselJ are special functions – the gamma function is the generalization of the factorial to non-integers, and the Bessel function is the solution to a common differential equation. Mupad has lots of built in special functions, which can be very useful. Notice also that, unlike MATLAB, Mupad returns the correct answer for sin(PI). This is because by default, Mupad is not working with floating point numbers. It will return the *exact* answer to any calculation. It will only start using floating point calculations if you start first, or explicitly ask for a numerical value. For example, contrast

```
[ gamma(1/2)
   $\sqrt{\pi}$ 
  gamma(0.5)
  1.772453851
```

In the second case, Mupad gives a floating point number because you typed in a floating point number (0.5) as the argument to the Gamma function. You can also ask Mupad to compute a numerical value for an expression with the ‘float’ function

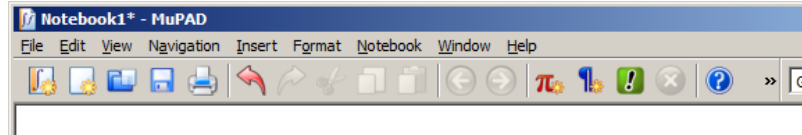
```
[ gamma(PI)
   $\Gamma(\pi)$ 
  float(%)
  2.288037795
```

Note the use of the % character – this always refers to the result of the last calculation that Mupad has done.

Note that unlike the MATLAB command window, Mupad lets you go back and change any line, and will then let you execute the file again with the changed code. The Notebook> menu gives lots of options for re-doing calculations after a correction.

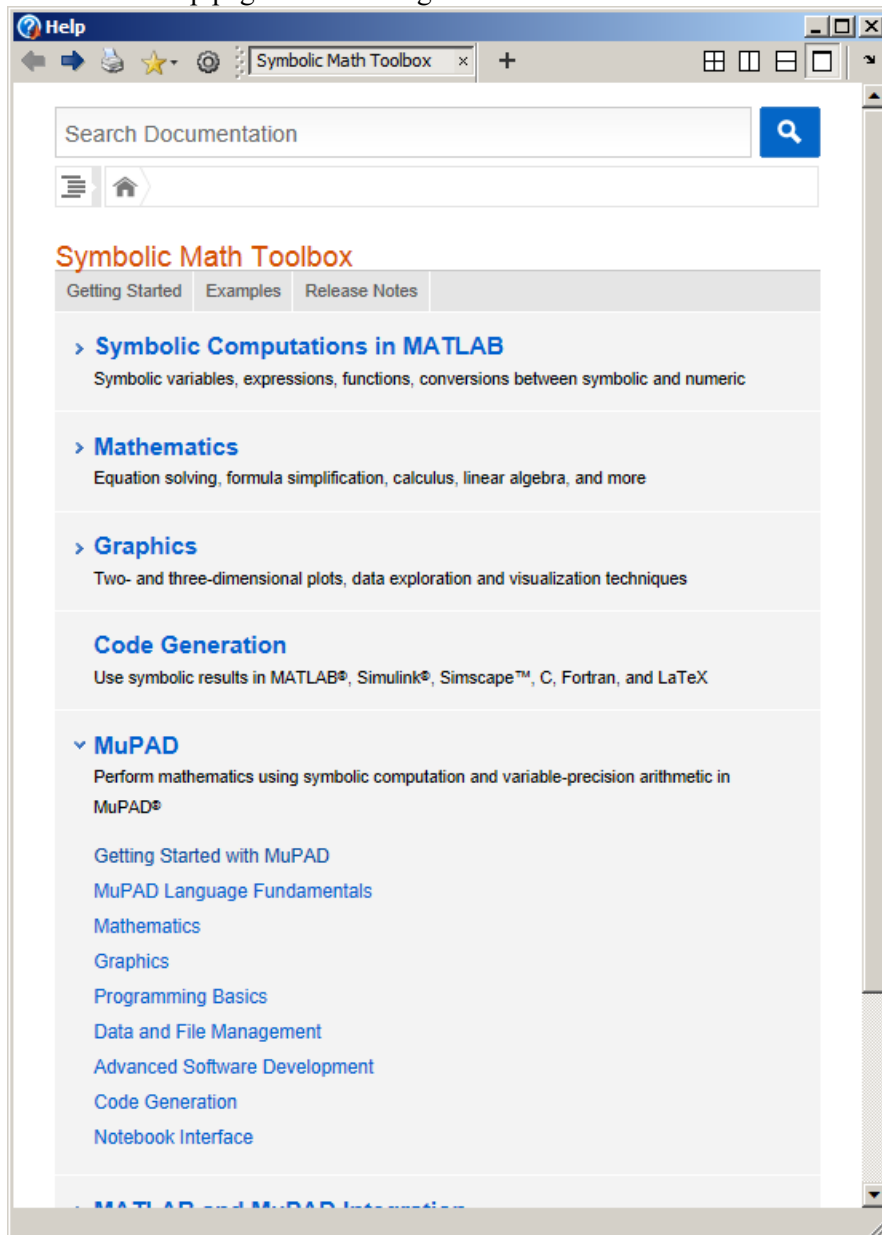
5. Help

Mupad will automatically open the help page for the MATLAB symbolic math toolbox. You can start help by pressing the question mark on the command ribbon or by going to Help, or by pressing the F1 key.





Note that the 'Search Documentation' box in the help window will search the whole MATLAB help, not just Mupad, so it's not very useful. But you can find a lot of Mupad examples and information by clicking on the 'Mupad' part of help.

Try browsing around in the help pages for a bit to get a sense of what is in there.



6. Saving your work

You can save your work in a Mupad 'Notebook' (a bit like a MATLAB script) by going to the File>Save menu. The file should be saved with the default .mn extension. Mupad is fairly robust, but it does crash unexpectedly now and again (usually when you try to resize a window), so it's worth saving lengthy calculations frequently.

Mupad notebooks can also include detailed comments and annotations that help readers follow what the calculations are doing. To insert a text paragraph, just hit the  button, or use Insert>Text Paragraph. Use Insert> Calculation to go back to typing in math, or use the  button.

You can also export a Mupad notebook to html or pdf format, if you want to publish your work.

7. Basic algebra

Mupad is quite good at doing algebra. For example, it can solve equations

```
eq1 := a*x^2 + b*x+c = 0
ax^2+bx+c=0
solve(eq1,x)
{
  { -b+sqrt(b^2-4ac), -b-sqrt(b^2-4ac) } if a ≠ 0
  { -c/b } if a=0 ∧ b ≠ 0
  C if a=0 ∧ b=0 ∧ c=0
  ∅ if a=0 ∧ b=0 ∧ c ≠ 0
}
solution := solve(eq1, x, IgnoreSpecialCases)
{ -b+sqrt(b^2-4ac), -b-sqrt(b^2-4ac) }
solution[1]
- (b+sqrt(b^2-4ac))/2a
solution[2]
- (b-sqrt(b^2-4ac))/2a
```

Because there are two solutions, they are returned in a set (enclosed by {}). You can extract each one by using the [number] convention.

IMPORTANT: Notice that the 'equals' sign is used in two different ways. If you just type $a=b$, you have created an equation object that you can use in later manipulations (e.g. solve it!). On the other

hand, if you type $a := b^2$ (with a colon) then you have assigned the value b^2 (a symbol) to a variable called a . Mupad will substitute b^2 for a any time it is used later. For example try this

```
[ a := b^2
  b^2
  eq1
  b^2 x^2 + b x + c = 0
  solution[1]
  - (b - sqrt(b^2 - 4 b^2 c)) / (2 b^2)
```

Notice that a in the 'eq1' object has been replaced by b^2 . You can clear the value of a variable using the 'delete' function

```
[ delete(a)
  a
  a
  solution[1]
  - (b + sqrt(b^2 - 4 a c)) / (2 a)
```

If you want to clear *all* variables, you can use the 'reset' function. This completely restarts mupad from the beginning. This is often useful for starting a new homework problem.

Let's try some more algebra

```
[ (x+2*y)^2
  (x+2 y)^2
  expand(%)
  x^2 + 4 x y + 4 y^2
  (x^2-y^2)/(x+y)
  (x^2-y^2) / (x+y)
```

Mupad doesn't simplify expressions by default. But it can do so if you ask it to

```
[ (x^2-y^2) / (x+y)
  simplify(%)
  x-y
```

This sort of thing is especially handy for trigonometric functions

```

[ sin(a+b+c)
  sin(a+b+c)
[ expand(%)
  cos(a) cos(b) sin(c) + cos(a) cos(c) sin(b) + cos(b) cos(c) sin(a)
  - sin(a) sin(b) sin(c)
[ simplify(%)
  sin(a+b+c)

```

Mupad can solve systems of equations too

```

[ reset
  reset
[ eq1 := a*x + b*y = c
  a x + b y = c
[ eq2 := c*x+d*y=e
  c x + d y = e
[ sol := solve({eq1,eq2},{x,y}, IgnoreSpecialCases)
  { [ x = - (b e - c d) / (a d - b c), y = (a e - c^2) / (a d - b c) ] }

```

You often want to solve an equation or system of equations, and then substitute that solution into a third equation. You can use the 'subs' function to do this

```

[ eq3 := x^3-y
  x^3 - y
[ subs(eq3,sol[1])
  - (a e - c^2) / (a d - b c) - (b e - c d)^3 / (a d - b c)^3
[ subs(eq3,sol[1][1])
  - y - (b e - c d)^3 / (a d - b c)^3
[ subs(eq3,sol[1][2])
  x^3 - (a e - c^2) / (a d - b c)

```

All the [1]s and [2]s here are hard to understand. Their purpose to extract the solutions from the variable 'sol.' Notice that 'sol' is in curly parentheses {} (look at the example at the top of the page) – this means sol contains a set (which happens to contain only a single solution – but in more complicated problems there might be more than one solution). You need to extract the solution you want out of this set. Thus, sol[1] extracts the first (and only) element from the set. In the first example, both the solutions for x and y are extracted and substituted into eq3. In the second example, sol[1][1] substitutes only x. In the third, sol[1][2] substitutes only y.

Of course not all equations can be solved exactly.

```
[-  
  solve(cosh(x)=2*x, x)  
  solve(cosh(x) - 2*x = 0, x)  
]
```

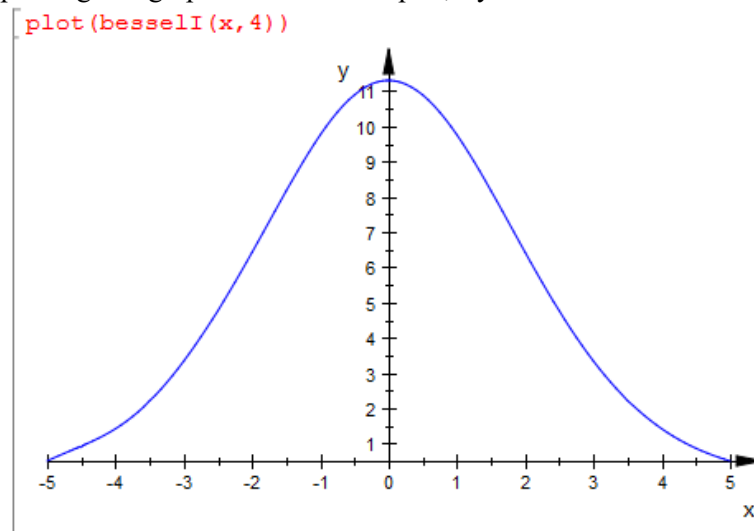
But you can get an approximate, numerical solution

```
[numeric::solve(cosh(x)=2*x, x)  
  {0.5893877635}  
  eqs := [x^2 = sin(y), y^2 = cos(x)]:  
  solution := numeric::fsolve(eqs, [x, y])  
  [x = -0.8517004887, y = 0.8116062151]  
]
```

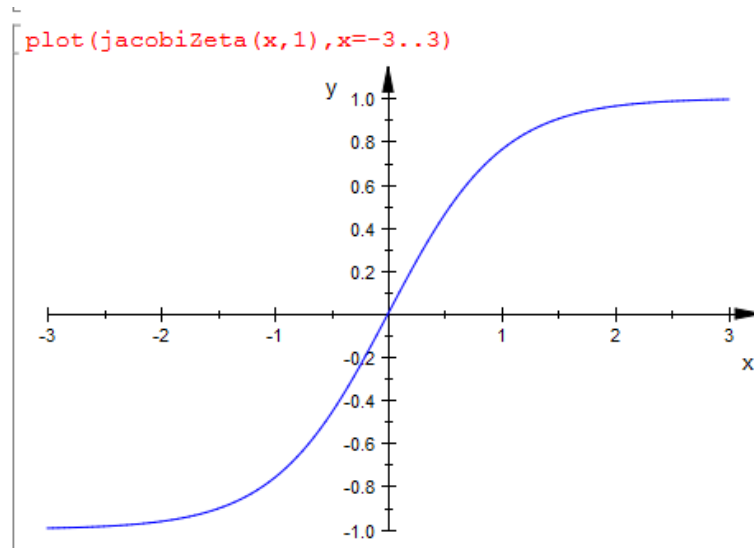
You can find more information about equation solving in the Mupad help documentation...

8. Plotting

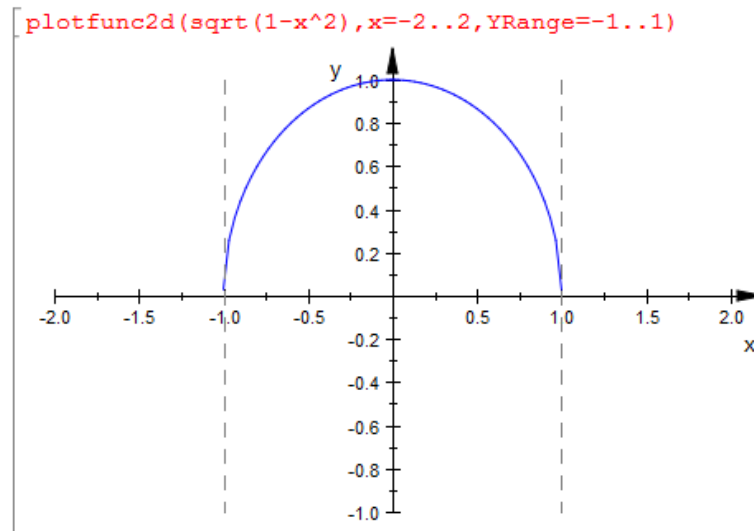
Mupad is very good at plotting and graphics. For a basic plot, try



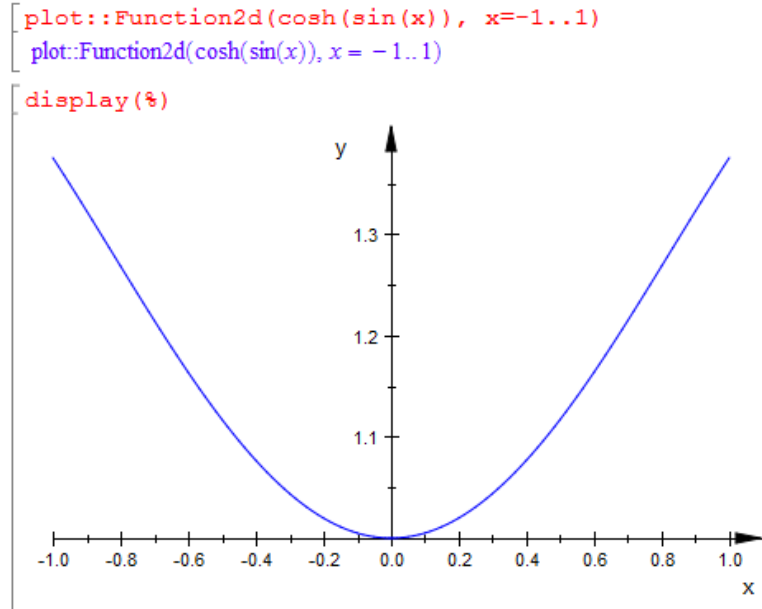
You can control the range of the plot as follows



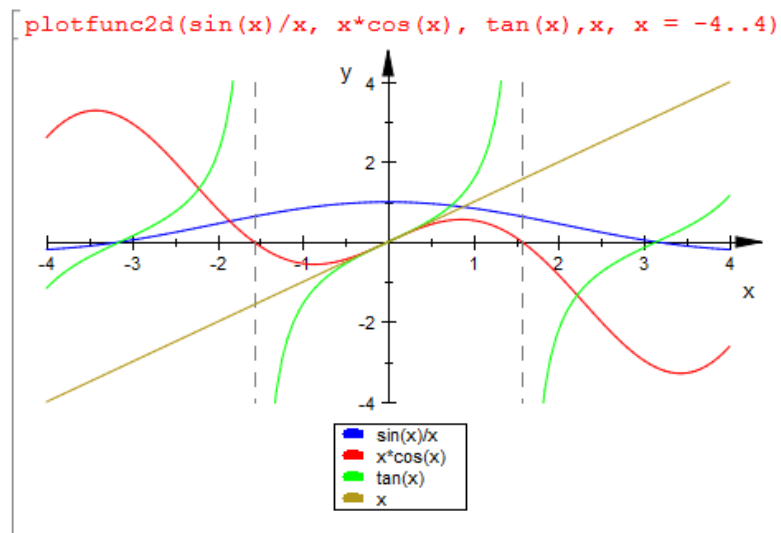
The 'plotfunc2d' command does the same thing as plot but has more options to control the appearance of the plot



Another way to do a plot is to select Plot Commands>Function Plots>2D Function from the menu on the right. The command will appear in the mupad window – you need to put in a function and range to replace the dummy arguments #f and #x=#a..#b . You will find the plot appears to do nothing. To display the plot you have to enter 'display(%)' on the next line

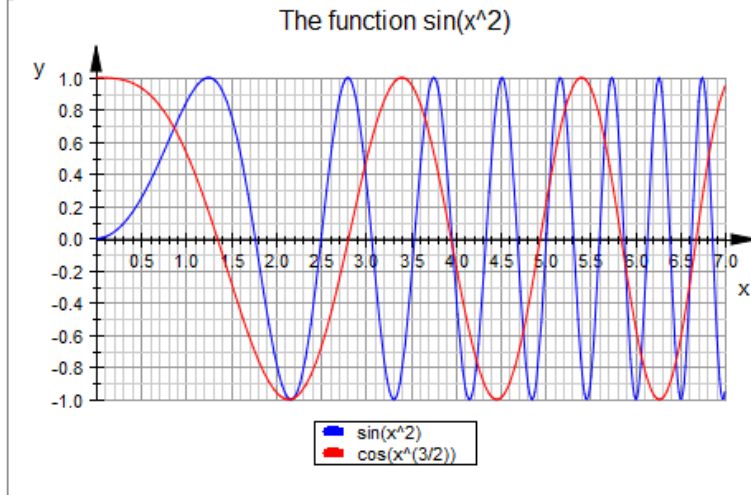


You can display multiple plots on the same axes



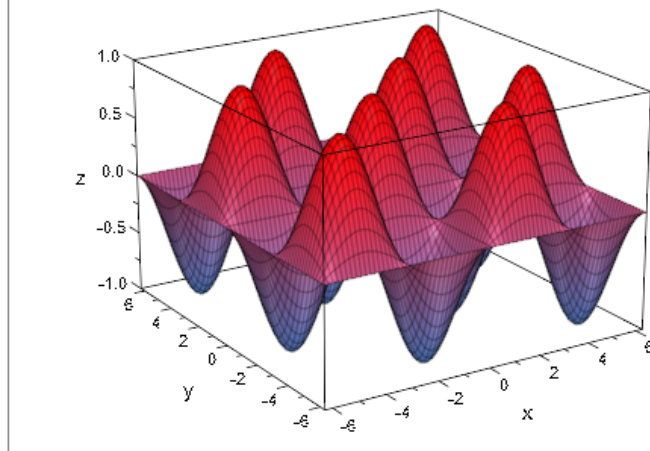
If you have no life and love to read help manuals, you make very fancy looking plots

```
plotfunc2d(sin(x^2),cos(x^(3/2)), x = 0..7,
Header = "The function sin(x^2)",
XTicksDistance = 0.5, YTicksDistance = 0.2,
XTicksBetween = 4, YTicksBetween = 1,
GridVisible = TRUE, SubgridVisible = TRUE):
```



(I do have a life – although there may not be much of it left - and hate to read manuals, so I just copied and pasted an example directly from the mupad help). You can do pretty 3D plots as well

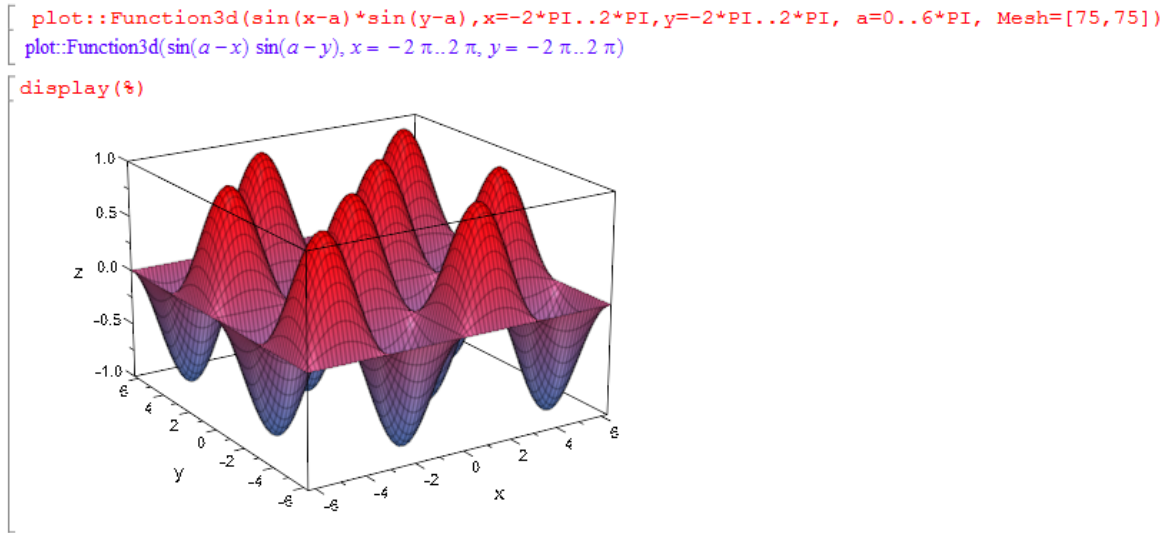
```
plot::Function3d(sin(x)*sin(y), x=-2*PI..2*PI, y=-2*PI..2*PI, Mesh=[75,75])
plot::Function3d(sin(x) sin(y), x = -2 π..2 π, y = -2 π..2 π)
display(%)
```



Click on the plot, and then try experimenting with some of the buttons on the toolbar window – you can rotate the plot around, zoom in, and so on.

You can make animations as well, by adding a 3rd parameter to a 3D plot (a in the example below). To play the animation, click on the picture, then press the big blue right pointing arrow.



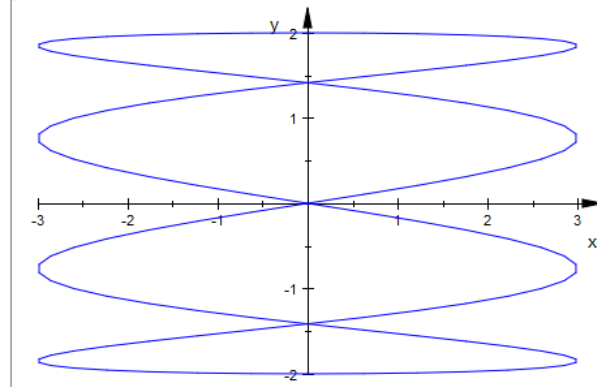


Mupad can do parametric plots as well, in both 2D and 3D. Try this

```

plot::Curve2d([3*sin(4*q),2*cos(q)], q=0..2*PI)
plot::Curve2d([3 sin(4 q), 2 cos(q)], q = 0..2 pi)
display(%)

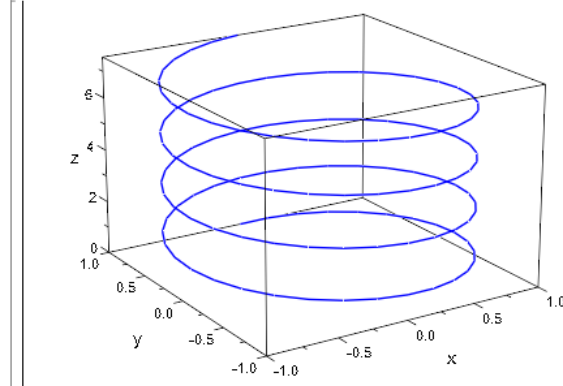
```



```

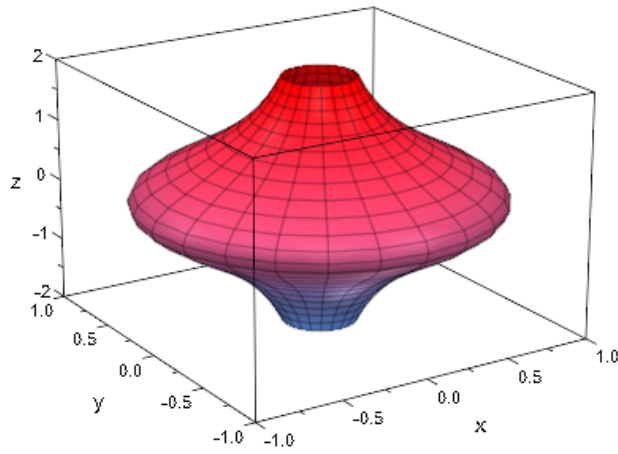
plot::Curve3d([sin(t),cos(t),0.3*t], t=0..a, a=0..8*PI)
plot::Curve3d([sin(t), cos(t), 0.3 t], t = 0..a)
display(%)

```



(The second plot here is an animation – you have to click on the plot to start the animation). You can plot 3D surfaces as well

```
plot::Surface([sin(t)/(1+u^2),cos(t)/(1+u^2),u], t=0..2*PI,u=-2..2)
plot::Surface([ $\frac{\sin(t)}{u^2+1}$ ,  $\frac{\cos(t)}{u^2+1}$ , u], t=0..2 pi, u=-2..2)
display(%)
```

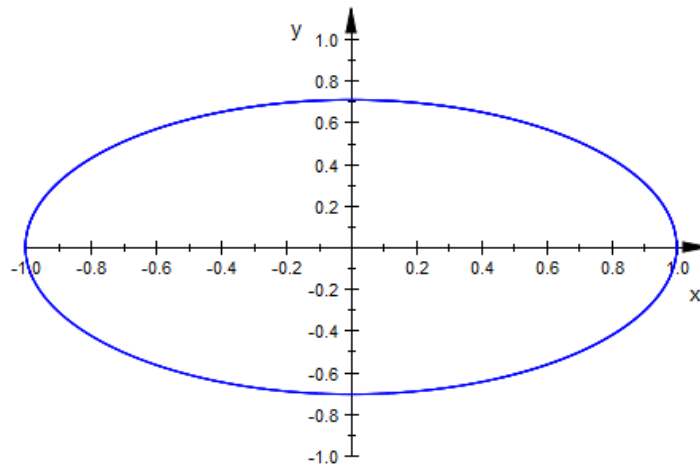


The 'implicitplot' is another very useful function. In 2D, it will plot a line or curve that satisfies an equation. In 3D, it will plot a plane or surface that satisfies a 3D equation. Here are two simple examples.

```
plot::Implicit2d(x^2+2*y^2=1, x=-1..1, y=-1..1)
```

```
plot::Implicit2d(x^2+2*y^2-1, x = -1..1, y = -1..1)
```

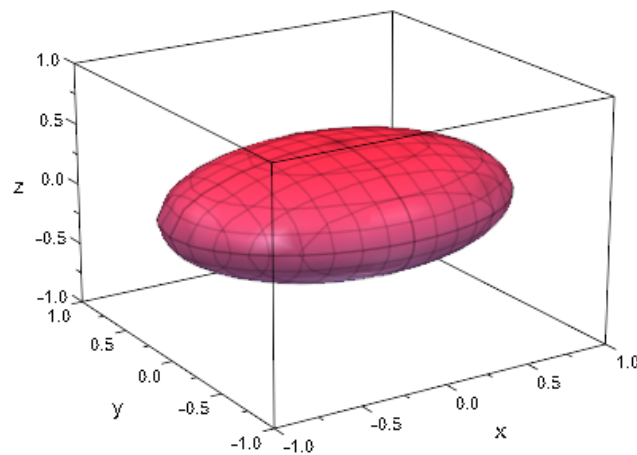
```
display(%)
```



```
plot::Implicit3d(x^2+2*y^2+4*z^2=1, x=-1..1, y=-1..1, z=-1..1)
```

```
plot::Implicit3d(x^2+2*y^2+4*z^2-1, x = -1..1, y = -1..1, z = -1..1)
```

```
display(%)
```



Saving plots: If you would like to include a plot in a report, you can click on the figure, then use Edit>Copy Graphics, then paste the figure into your document. You can also export the figure to a file in various formats (you will need to do this to save animations) using File>Export Graphics.

9. Calculus

Mupad is great at calculus. Try

```
[diff(x^2,x)
 2 x
int(%,x)
x^2
diff(x*sin(x)^2*cos(x)^2,x)
2 x cos(x)^3 sin(x) + cos(x)^2 sin(x)^2 - 2 x cos(x) sin(x)^3
simplify(%)
x sin(4 x) / 2 - cos(4 x) / 8 + 1 / 8
int(%,x)
x sin(2 x)^2 / 4
expand(%)
x cos(x)^2 sin(x)^2
```

Mupad can do partial derivatives as well

```
[f:=sqrt(x^2+y^2+z^2)
sqrt(x^2+y^2+z^2)
diff(f,x)
x / sqrt(x^2+y^2+z^2)
```

It can also do definite integrals

```
[assume('sigma;'>0)
f := exp(-x^2/'sigma;'^2)
e^(-x^2/sigma^2)
int(f,x=-infinity..infinity)
sigma*sqrt(pi)
f:= x*log(x)
x ln(x)
int(f,x=a..b)
b^2*(ln(b)-1/2) / 2 - a^2*(ln(a)-1/2) / 2
simplify(%)
a^2/4 - b^2/4 - a^2*ln(a)/2 + b^2*ln(b)/2
```

Try the integral without the ‘assume(‘σ’>0) as well (just say delete(‘σ’) and do the integral again – you get a big mess). Notice also that Mupad interprets log(x) to be the *natural* log – this is standard practice in math and engineering (log₁₀ very rarely comes up except in signal processing e.g. to define things like decibels).

Of course not all integrals can be evaluated... But definite integrals can always be evaluated numerically.

```

[ f := sin(sin(x))
  sin(sin(x))
  int(f,x=0..PI/2)
  ∫0π/2 sin(sin(x)) dx
  float(%)
  0.893243741

```

Another very useful application is to take limits and Taylor series expansions of functions

```

[ limit(sin(x)/x, x=0)
  1
  taylor(sin(x)/x, x)
  1 - x2/6 + x4/120 + O(x6)
  expr(%)
  x4/120 - x2/6 + 1
  taylor(cos(x)/(1-x), x=0, 8)
  1 + x + x2/2 + x3/2 + 13x4/24 + 13x5/24 + 389x6/720 + 389x7/720 + O(x8)

```

Here, the ‘expr(%)’ gets rid of the funny $O(x^6)$ that denotes how many terms were included in the series – this can be useful if you want to substitute the Taylor expansion into another equation later.

Mupad will also sum series for you – but we won’t need that much in EN40. You can explore it for yourself if you are curious.

10. Finding maxima and minima

You can use Mupad to do the usual calculus to maximize or minimize a function. For example, let’s try to find the maximum value of

$$f(x) = x^4 \exp(-x)$$

To do this by hand, you would (i) find the values of x that make f stationary, i.e. solve $df/dx = 0$, and (ii) substitute the solution(s) back into the function. Here’s the Mupad version of this

```

[ reset() :
[ f := x^4*exp(-x) :
[ xatmax := solve(diff(f,x)=0,x)
[ {0,4}
[ subs(f,x=xatmax[2])
[ 256 e^-4

```

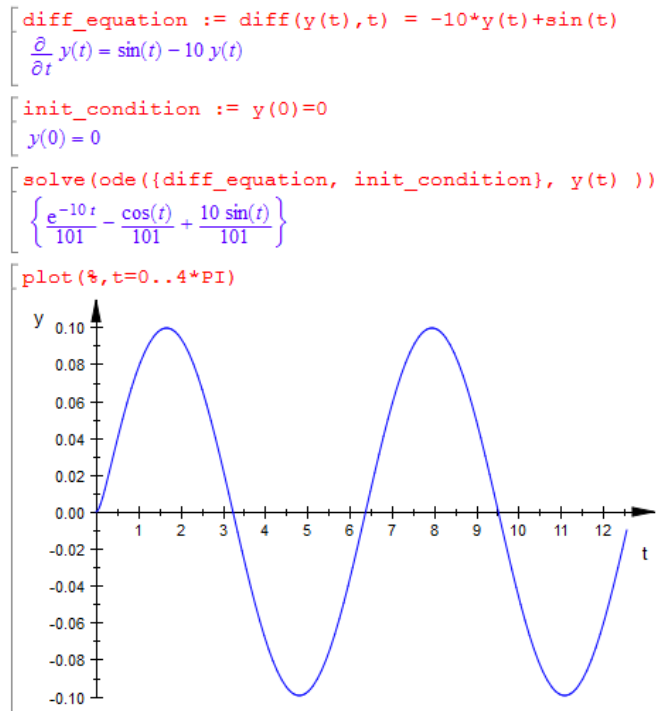
Notice that Mupad gives all the roots of $df/dx=0$. We have to work out which one corresponds to a maximum. You can do this by plotting the function, or in this case we can see that $x=0$ has to be a minimum (since $f>0$ everywhere else!). The statement `subs(f,x=xatmax[2])` substitutes the second root back into f .

11. Solving differential equations

Mupad can also solve differential equations – both analytically, and numerically (but in this course we will use MATLAB whenever we want a numerical solution). For example, let's solve the differential equation from the MATLAB tutorial:

$$\frac{dy}{dt} = -10y + \sin(t)$$

given that $y=0$ at time $t=0$.



Notice that Mupad gives a *formula* for the solution (recall that MATLAB only gives numbers). Here's another example – this is the differential equation governing the free vibration of a damped spring-mass system, which will be discussed in painful detail later in the course

```
diff_equation := y''(t) + 2*`&zeta;`*`&omega;`*y'(t) + `&omega;`^2*y(t)
y''(t) + ζ y'(t) ω 2 + y(t) ω² = 0

init_condition := y(0)=y0, y'(0)=v0
y(0) = y0, y'(0) = v0

solve(ode({diff_equation, init_condition}, y(t) ), IgnoreSpecialCases)
{
  e^{r(ωσ₁ - ωζ)} (v0 + ωζy0 + ωy0σ₁) - e^{-r(ωσ₁ + ωζ)} (v0 + ωζy0 - ωy0σ₁)
  / (2ωσ₁)
}

where
σ₁ = √((ζ - 1)(ζ + 1))

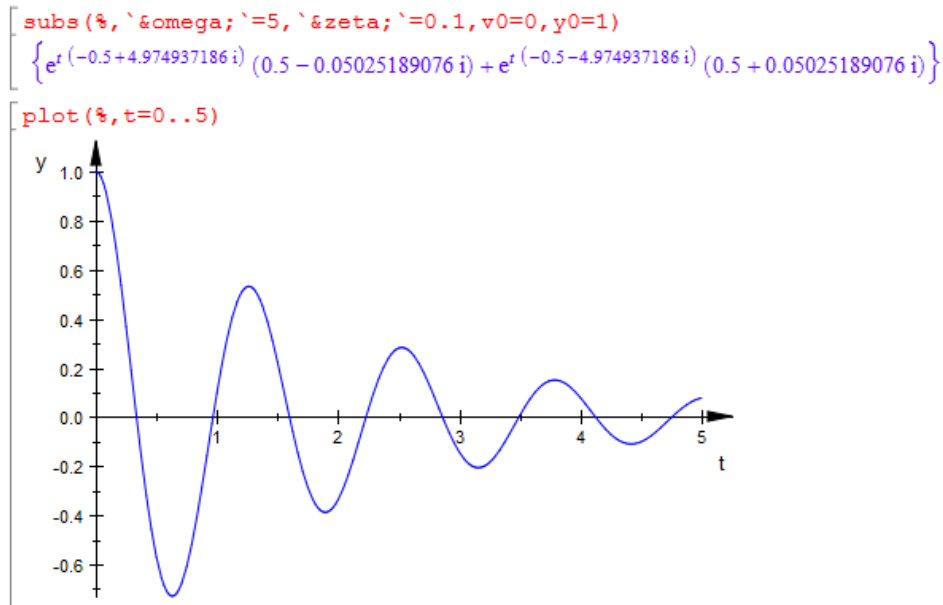
simplify(%)
{
  e^{-ωt(ζ - σ₁)} (v0 + ωζy0 + ωy0σ₁) - e^{-ωt(ζ + σ₁)} (v0 + ωζy0 - ωy0σ₁)
  / (2ωσ₁)
}

where
σ₁ = √(ζ² - 1)
```

Here, we used the notation

$$y''(t) \equiv \frac{d^2 y}{dt^2} \quad y'(t) \equiv \frac{dy}{dt}$$

Again, mupad gives an exact solution – but it's not very easy to visualize what the solution looks like! If we substitute numbers we can plot it



Of course, not all differential equations can be solved exactly.

```

diff_equation := y(t)*y'(t) + 2*`&zeta;`*`&omega;`*y'(t) + `&omega;`^2*y(t)=0
y''(t) y(t) + ζ y'(t) ω 2 + y(t) ω² = 0

init_condition := y(0)=y0, y'(0)=v0
y(0) = y0, y'(0) = v0

solve(ode({diff_equation, init_condition}, y(t) ), IgnoreSpecialCases)
solve(ode({y(0) = y0, y'(0) = v0, y''(t) y(t) + ζ y'(t) ω 2 + y(t) ω²}, y(t)), IgnoreSpecialCases)

```

Mupad now gives no solution. It is possible for Mupad to compute a numerical approximation, but in most cases it is more straightforward to use MATLAB if an analytical solution cannot be found. We will use MATLAB exclusively for this purpose in this course.

11. Vectors and Matrices

Finally, we'll take a look at Mupad's functions that deal with vectors and matrices. Strangely, vector and matrix manipulations are more cumbersome than most of Mupad's capabilities – even doing a trivial thing like a dot product is a chore. Here are some basic vector/matrix manipulations.

```

reset
reset

vcol := matrix([x, y, z])
⎛ x ⎞
⎜ y ⎟
⎝ z ⎟

vrow := matrix([[w, s, t]])
( w s t )

Mx := matrix([[a, b, c], [d, e, f], [g, h, i]])
⎛ a b c ⎞
⎜ d e f ⎟
⎝ g h i ⎟

Mx*vcol
⎛ ax+by+cz ⎞
⎜ dx+ey+fz ⎟
⎝ gx+hy+iz ⎟

vrow*%
( w(ax+by+cz) + s(dx+ey+fz) + t(gx+hy+iz) )

linalg::scalarProduct(vrow, vcol, Real)
s y + t z + w x

linalg::scalarProduct(vrow, vrow)
s  $\bar{s}$  + t  $\bar{t}$  + w  $\bar{w}$ 

```

These commands create row and column vectors; a matrix' multiply a matrix by a vector (to produce a column vector); multiply a column vector by a row vector to produce a scalar, and do a dot product of two vectors (the scalar product) in two different ways. Note that by default Mupad assumes that all variables could be complex numbers – hence the complex conjugates unless you specify 'Real' in the ScalarProduct function.

Mupad can do fancy things like cross products and vector calculus (curl, divergence, etc). It can even find the scalar potential corresponding to a curl free vector field (useful for calculating the potential energy of a force, for example). It can find a vector potential for a divergence free vector field too (not so useful for us here...)

```

[ linalg::crossProduct(vrow,vcol)
  (s z - t y t x - w z w y - s x)
[ vfun := matrix([vx(x,y,z),vy(x,y,z),vz(x,y,z)])
  ( vx(x,y,z)
    vy(x,y,z)
    vz(x,y,z)
[ divergence(vfun,[x,y,z])
   $\frac{\partial}{\partial x} vx(x,y,z) + \frac{\partial}{\partial y} vy(x,y,z) + \frac{\partial}{\partial z} vz(x,y,z)$ 
[ curl(vfun,[x,y,z])
  (  $\frac{\partial}{\partial y} vz(x,y,z) - \frac{\partial}{\partial z} vy(x,y,z)$ 
     $\frac{\partial}{\partial z} vx(x,y,z) - \frac{\partial}{\partial x} vz(x,y,z)$ 
     $\frac{\partial}{\partial x} vy(x,y,z) - \frac{\partial}{\partial y} vx(x,y,z)$ 
[ r := sqrt(x^2+y^2+z^2)
   $\sqrt{x^2+y^2+z^2}$ 
[ gravityforce := matrix([k*x/r^3,k*y/r^3,k*z/r^3])
  (  $\frac{kx}{(x^2+y^2+z^2)^{3/2}}$ 
     $\frac{ky}{(x^2+y^2+z^2)^{3/2}}$ 
     $\frac{kz}{(x^2+y^2+z^2)^{3/2}}$ 
[ potential(gravityforce,[x,y,z])
   $-\frac{k}{\sqrt{x^2+y^2+z^2}}$ 

```

This should be enough to get you started, but we've only looked at a small subset of Mupad's capabilities. You might like to explore the help manual to find more obscure tricks. Mupad can be useful in many of your courses, not just EN40!

COPYRIGHT NOTICE: This tutorial is intended for the use of students at Brown University. You are welcome to use the tutorial for your own self-study, but please seek the author's permission before using it for other purposes.

A.F. Bower
School of Engineering
Brown University
December 2016