Symbolic Maths v2

Dr Amir-Homayoun Javadi
a.h.javadi@gmail.com
www.javadilab.com

MATLAB – Symbolic Math Toolbox	Python – SymPy Toolbox
power/exponentiation	
^ and **	**
defining variables	
a = sym('a') syms x h - syms is the shortcut for sym	a = Symbol('a') [x, h] = Symbol(['x', 'h']) x, h = symbols('x, h') # an alternative - Pay attention to small and capital letters!
assumptions	
syms x assume(x >= 0) assumeAlso(x, 'integer') assume(x,'clear') assumptions(x) - For list of assumptions see this link.	x = Symbol('x') a, b = symbols('a, b', nonnegative=true) x = Symbol('x', integer=True) x = Symbol('x', clear=True) x.assumptions0 - For list of assumptions see this link.
list of variables	

syms a x y
f = a * x ^ 2 + y
symvar(f, 1)
  = x% the default variable
syms(f, 3)
  = [a, x, y]
	a, x, y = symbols('a, x, y')
	f = a * x ** 2 + y
	# nothing equivalent
	
	f.free_symbols
	  = {a, x, y}

create symbolic functions	

syms f(x, y)
f(x, y) = x + y
f(1, 2)
  = 3
syms x y
f = symfun(x + y, [x y])
	x, y = symbols('x, y')
	f = x + y
	f.subs([(x, 1), (y, 2)])
	  = 3

substitute and evaluate functions	

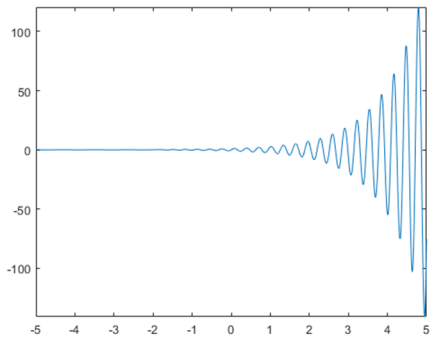
syms x y
f = x + y
f = subs(f, x, 2) % substitutes x with 2
  = 2 + y
 	
y = x^2
x = 2
subs(y)
  = 4
eval(y)
  = 4
- pay attention that now y is not the x^2 formula anymore.
	x, y = symbols('x, y')
	f = x + y
	f.subs(x, 2) # substitutes x with 2
	  = 2 + y
	
	y = x ** 2
	y.evalf(subs={x: 2})
	  = 4.0000

defining equations	
eqn1 = sin(x)+y == x^2 + y^2 - eqn1 is name of the equation. - == sign is for assignment in this case (not condition).	eqn1 = Eq(sin(x)+y, x**2 + y**2)
solving equations	

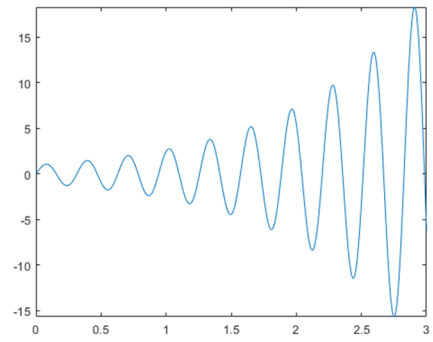
syms x y a b
f(x) = x – 5 * a + b
solve(f) % solves for x
  = 5 * a – b
solve(f, a) % solves for a
  = b/5 + x/5
 	
eqn = x+y == 2*x + y^2
solve(x)
  = y – y^2
	a, x, y = symbols('a, x, y')
	f = x – 5 * a + b
	solve(f, x)
	  = [5*a - b] # solves for x
	solve(f)
	  = [{a: b/5 + x/5}] # solves for a
	
	eqn = Eq(x+y, 2*x+y**2)
	solve(eqn)
	  = [{x: y*(-y + 1)}]

plotting	
syms t x y	from sympy.plotting import * t, x, y = symbols('t, x, y')

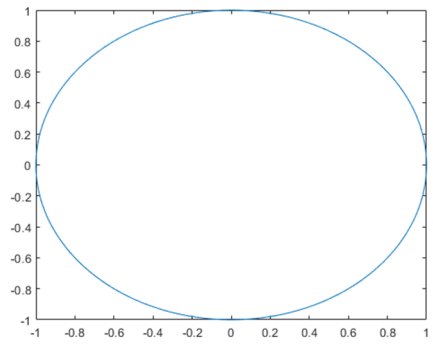
```
f = exp(x)*sin(20*x)
fplot(f)
```



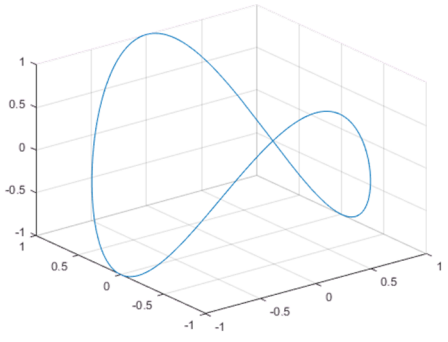
```
fplot(f,[0 3])
```



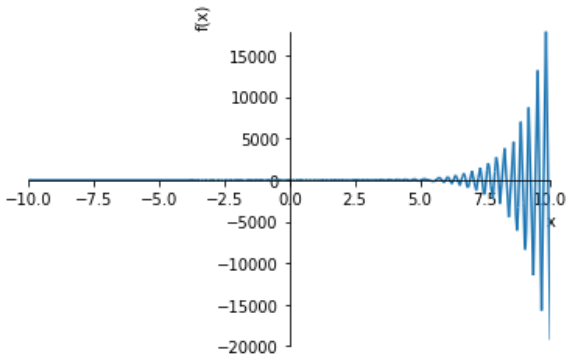
```
fplot(cos(x), sin(x))
```



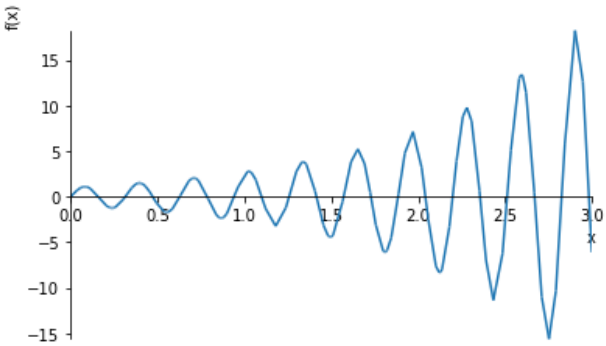
```
fplot3(sin(t), cos(t), cos(2 * t))
```



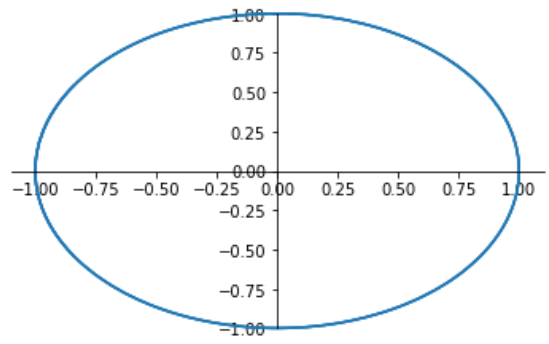
```
f = exp(x)*sin(20*x)
plot(f)
```



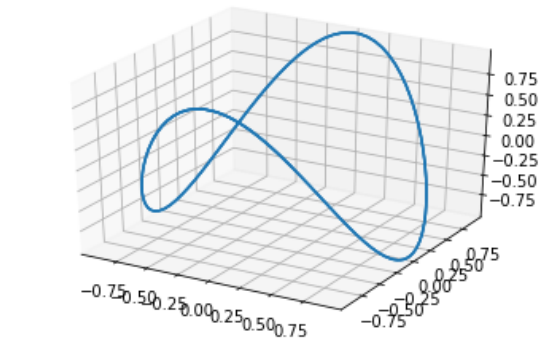

```
plot(f, (x, 0, 3))
```



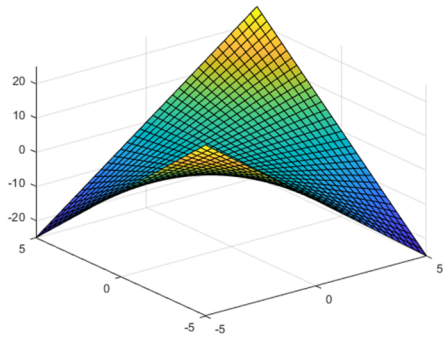

```
plot_parametric(cos(x), sin(x))
```



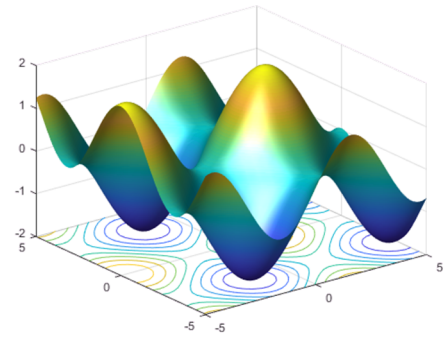
```
plot3d_parametric_line(sin(t), cos(t), cos(2 * t))
```



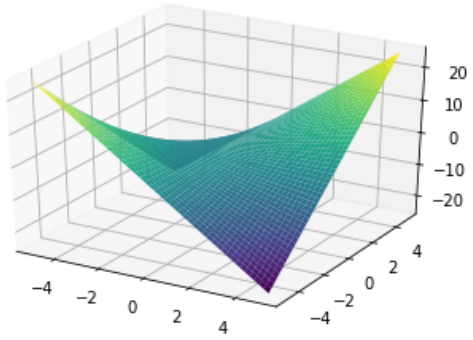
```
fsurf(x * y)
```



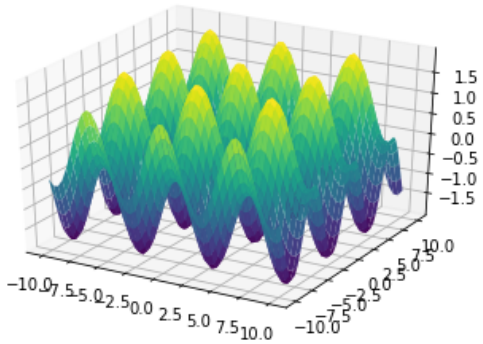
```
fsurf(sin(x) + cos(y),
    'EdgeColor', 'none',
    'ShowContours', 'on')
camlight(110, 70)
```



```
plot3d(x*y, (x, -5, 5), (y, -5, 5))
```



```
plot3d(sin(x) + cos(y))
```



Note

1. To use SymPy in Python you need to import SymPy to your code, for example using the line below:

```
from sympy import *
```

2. Anaconda package contains SymPy. But if you don't have SymPy installed on your machine, please refer to my How to Install Python Toolboxes handout.
3. One of the best options for working with symbolic math toolbox in MATLAB is Live Editor. This is an option for MATLAB v2016 and later. There is "New Live Script" button in Home tab of the ribbon menu.
4. When you run Symbolic Math in MATLAB in Live Script the equations are displayed nicely. But in Python it doesn't. If you want to get a better display you need to use `pprint` (pretty print) command:

```
pprint(x ** 2 + x / 2)
      2   x
= x  + ─
       2
```