Author: Brian Mungo
Email: mungoengineering@gmail.com

03/02/2018
Rev 1

SOLVE A SECOND ORDER DIFFERENTIAL EQUATION WITH GIVEN INITIAL CONDITIONS USING SYMPY

This discussion will solve the following differential equation (DE) with given initial conditions using a Python module called Sympy.

$$\ddot{y} + \dot{y} + y = 0\,;\, y(0) = 1\,;\; \dot{y}(0) = 0 \qquad (1)$$

Step 1: Import all modules and define the independent variable ‘t’.

```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp
from sympy.abc import t
```

Step 2: Define your dependent variable in symbol form [2].

```
y = sp.symbols('y',cls=sp.Function)
```

Step 3: Using Sympy.dsolve solve the DE.

```
y1 = sp.dsolve(y(t).diff(t,t) + y(t).diff(t) + y(t),y(t))
```

Output:

```
Eq(y(t), (C1*sin(sqrt(3)*t/2) + C2*cos(sqrt(3)*t/2))/sqrt(exp(t)))
```

The output above is the solution to (1). The output looks confusing however and could use some context before moving forward. y1’s data type is called a ‘sympy.core.relational.Equality’ [2]. That’s a fancy term for stating that the 1st index of y1 is equal to the 2nd index. Or,

$$y(t) = C1sin \ldots$$

If we type the following,

```
print y1.lhs
print y1.rhs
```

Output:

```
y(t)
(C1*sin(sqrt(3)*t/2) + C2*cos(sqrt(3)*t/2))/sqrt(exp(t))
```

The (.lhs) means Left Hand Side, while (.rhs) means Right Hand Side.

Step 4: Solve the constants using the given initial conditions. I have not found an easy way to do this yet. Apparently, this concept is not fully implemented yet. The following seems to work well though [3]. To solve for the constants, we must solve a system of equations by setting the value of the independent variable and bringing over the equality condition to one side. The first index below is the equation y1 equal to one, while the second is y1's derivative equal to zero. [Do not put the = sign in; Move to one side]

```
C = sp.solve([y1.rhs.subs(t,0) - 1.0, y1.rhs.diff(t).subs(t,0) - 0.0])
```

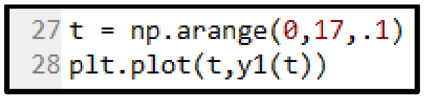
Step 5: Substitute the constants back into y1.

```
y1 = y1.subs(C)
```

Step 6: Turn the y1 function into an equation that can be numerically solved [3]. This will allow you to graph y1 in terms of an array of independent values. I do not know why 'numpy' is used below.

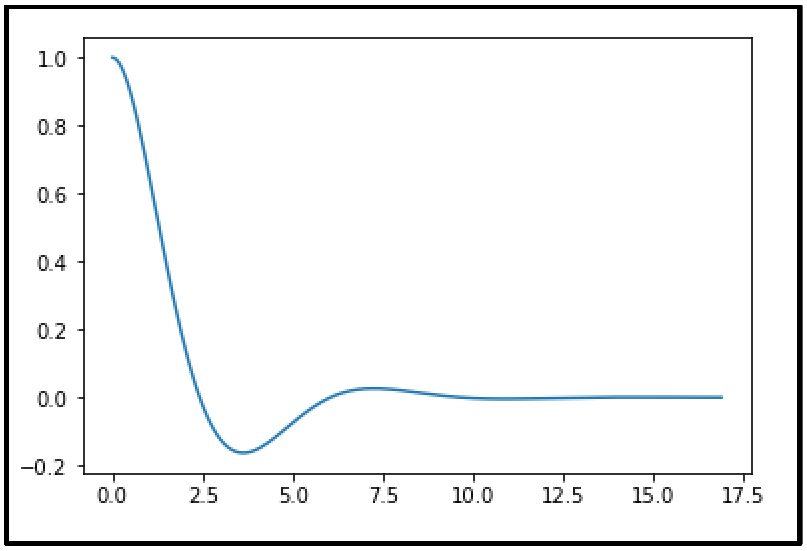
```
y1 = sp.lambdify(t , y1.rhs, 'numpy')
```

Step 7: Create an array of 't' values and plot y1 in terms of 't' using matplotlib.pyplot.



```
t = np.arange(0,17,.1)
plt.plot(t,y1(t))
```

Output:



References:

[1] http://docs.sympy.org/latest/index.html

[2] http://docs.sympy.org/latest/modules/solvers/ode.html

[3] http://www.eg.bucknell.edu/physics/ph329/jupyter/ode_sympy.ipynb.pdf