06_assignment_odes

Debaraj Barua

May 21, 2017

1 Hochschule Bonn-Rhein-Sieg

2 Mathematics for Robotics and Control, SS17

3 Assignment 6: Ordinary Differential Equations

Familiarize yourself with SymPy:

SymPy Tutorial

In particular, look at how to solve differential equations with SymPy: Differential equations with SymPy

```
In [53]: import sympy as sp
         import numpy as np
         # Enable latex printing
         sympy.init_printing(use_latex=True)
```

For your first assignment, solve the following three ODEs *by hand* and verify your results using SymPy. Please write the steps of your manual solutions using TeX syntax in the input cell below. You can find an introduction to mathematical expressions in LaTeX here. In order for the IPython notebook to evaluate your mathematical expressions, enclose them in $. For instance,

```
$ \sum_{i=1}^{10} t_i $
```

will display the following expression: $\sum_{i=1}^{10} t_i$

3.0.1 1. Solve the following ordinary differential equations by hand and verify your results using SymPy [20]

Equation 1.1: $y' = 5 \cdot y$

Solve Equation 1.1 by hand and write your steps here using LaTeX

$$
\begin{aligned}
&\frac{dy}{dx} = 5 \cdot y \\
&\implies \frac{dy}{y} = 5 \cdot dx \\
&\implies \int \frac{dy}{y} = \int 5 \cdot dx \\
&\implies \ln y = 5 \cdot x + C \\
&\implies y = e^{5x+C} \\
&\implies y = e^{5x} \cdot e^{C} \\
&\implies y = C_1 \cdot e^{5x} \quad ✓
\end{aligned}
$$

```
In [9]: # Insert code to verify your solution for Equation 1.1 here and evaluate it

        x = sp.Symbol('x')
        y=sp.Function('y')

        y_prime=sp.Derivative(y(x),x)
        eq= y_prime - 5*y(x)
        sp.init_printing(use_latex=True)
        sp.dsolve(eq)
```

✓

```
Out[9]:
```

$$y(x) = C_1 e^{5x}$$

Equation 1.2: $\frac{dy}{dx} = -2 \cdot x \cdot y$
Solve Equation 1.2 by hand and write your steps here using LaTeX

$$
\begin{aligned}
&\frac{dy}{dx} = -2 \cdot x \cdot y \\
&\implies \frac{dy}{y} = -2 \cdot x \cdot dx \\
&\implies \int \frac{dy}{y} = \int -2 \cdot x \cdot dx \\
&\implies \ln y = -2 \cdot \frac{x^2}{2} + C \\
&\implies y = e^{-x^2+C} \\
&\implies y = e^{-x^2} \cdot e^{C} \\
&\implies y = C_1 \cdot e^{-x^2} \quad ✓
\end{aligned}
$$

```
In [10]: # Insert code to verify your solution for Equation 1.2 here and evaluate it
```

```
x = sp.Symbol('x')
y=sp.Function('y')

y_prime=sp.Derivative(y(x),x)
eq=y_prime+2*x*y(x)
sp.init_printing(use_latex=True)
sp.dsolve(eq)
```

Out[10]:

$$y(x) = C_1 e^{-x^2}$$ ✓

Equation 1.3: $\frac{dy}{dx} = y^2$
Solve Equation 1.3 by hand and write your steps here using LaTeX

$$\begin{aligned}
&\frac{dy}{dx} = y^2 \\
&\implies \frac{dy}{y^2} = dx \\
&\implies \int \frac{dy}{y^2} = \int dx \\
&\implies -\frac{1}{y} = x + C_1 \\
&\implies y = -\frac{1}{x + C_1}
\end{aligned}$$ ✓

In [11]: # Insert code to verify your solution for Equation 1.3 here and evaluate it

```
x = sp.Symbol('x')
y=sp.Function('y')

y_prime=sp.Derivative(y(x),x)
eq=y_prime-y(x)**2
sp.init_printing(use_latex=True)
sp.dsolve(eq)
```

Out[11]:

$$y(x) = -\frac{1}{C_1 + x}$$

Equation 1.4: $y' + \frac{4}{x} \cdot y = x^4$
Solve Equation 1.4 by hand and write your steps here using LaTeX
We have,

$$\frac{dy}{dx} + \frac{4}{x} \cdot y = x^4$$

Now, let us assume, a function $f(x)$ such that $\frac{df}{dx} = f(x) \cdot \frac{4}{x}$. So,

yup. you can directly use the equation for calculating the integrating factor though

$$
\begin{aligned}
&\frac{df}{dx} = f \cdot \frac{4}{x} \\
&\implies \frac{df}{f} = \frac{4}{x} \cdot dx \\
&\implies \int \frac{df}{f} = \int \frac{4}{x} \cdot dx \\
&\implies \ln f = 4 \cdot \ln x + C_1 \\
&\implies f = e^{4 \cdot \ln x + C_1} \\
&\implies f = \left(e^{\ln x}\right)^4 \cdot e^{C_1} \\
&\implies f = x^4 \cdot C_2
\end{aligned}
$$

Therefore our equation can be re-written as:

$$
\begin{aligned}
&f \cdot \frac{dy}{dx} + f \cdot \frac{4}{x} \cdot y = f \cdot x^4 \\
&\implies f \cdot \frac{dy}{dx} + \frac{df}{dx} \cdot y = f \cdot x^4 \\
&\implies \frac{d}{dx}(f \cdot y) = f \cdot x^4 \\
&\implies \int d(f \cdot y) = \int f \cdot x^4 dx \\
&\implies f \cdot y = \int f \cdot x^4 dx \\
&\implies y = \frac{\int f \cdot x^4 dx}{f}
\end{aligned}
$$

Using the equation $f(x) = x^4 \cdot C_2$,

$$
\begin{aligned}
&y = \frac{\int f \cdot x^4 dx}{f} \\
&\implies y = \frac{\int x^8 \cdot C_2 \cdot dx}{x^4 \cdot C_2} \\
&\implies y = \frac{\frac{x^9}{9} + C_3}{x^4} \\
&\implies y = \frac{C_3}{x^4} + \frac{x^5}{9} \quad ✓
\end{aligned}
$$

```
In [8]: # Insert code to verify your solution for Equation 1.4 here and evaluate it
```

```
x = sp.Symbol('x')
y=sp.Function('y')

y_prime=sp.Derivative(y(x),x)
eq=y_prime+4/x*y(x)-x**4
sp.init_printing(use_latex=True)
sp.dsolve(eq)
```

```
Out[8]:
```

$$y(x) = \frac{C_1}{x^4} + \frac{x^5}{9}$$ ✓

3.0.2 Numerical methods for solving differential equations [50]

In the lab class, you used Euler's method to numerically estimate the solution to an ODE; this method belongs to the family of Runge-Kutta methods. Here you will use the fourth-order Runge-Kutta method to solve an ODE numerically. The higher order Runge-Kutta methods are more accurate since they also consider the slopes at points between the current and next step (instead of just the slope at the current point).

The relevant equations needed are:

$$y' = f(x, y)$$

$$x_{n+1} = x_n + h$$

$$y_{n+1} = y_n + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

$$k_1 = h \cdot f(x_n, y_n)$$

$$k_2 = h \cdot f(x_n + \frac{h}{2}, y + \frac{k_1}{2})$$

$$k_3 = h \cdot f(x_n + \frac{h}{2}, y + \frac{k_2}{2})$$

$$k_4 = h \cdot f(x_n + h, y + k_3)$$

The context of the exercise is as follows: a robot is tasked with changing a lamp in a factory. The task involves pressing a switch to turn the lamp off, waiting until it cools down to 30° C, and then unscrewing it. The robot is able to query the temperature of the lamp as long as it is turned on, but once the lamp is turned off, the temperature cannot be queried.

Newton's law of cooling states that the rate of change of temperature of an object is directly proportional to the difference between the temperature of the object and the ambient temperature. This can be written in the form of a differential equation as follows:

$$\frac{dT}{dt} = -K(T - T_A)$$

where T is the temperature of the object, T_A is the ambient temperature and K is a constant that depends on the properties of the object.

The robot queries the temperature of the lamp just before turning it off and receives $T_{t=0} = 150°$ C. The ambient temperature is known to be $T_A = 20°$ C. The robot also knows that for this lamp, $K = 0.05$.

The robot needs to calculate how long it needs to wait until the lamp has cooled down sufficiently; i.e. find the t when the temperature is just below 30° C.

Solve this problem using fourth-order Runge-Kutta, using step sizes 1, 0.1 and 0.01.

```
In [54]: # Implement the Runge-Kutta method here to solve
         # the given differential equation

         K = 0.05
         #h = 0.01 # also show the result with h = 1 and 0.1
         T_A = 20.0
         stop_temp = 30.0
         start_temp = 150.0

         """
         To find t when T=30
         """

         """
         T_prime: Function gives the value of the derivative dT/dt
                  at Temperature "Temp" and time "t"
         """
         def T_prime(t, Temp):
             return -K*(Temp-T_A)


         """
         R_K : Function returns the time estimate using fourth-order Runge-Kutta
         ic  : Initial values ic[0]: time at t_0, and ic[1]: Temperature at t_0
         h   : Step size
         """
         def R_K(ic, h):

             t_current = ic[0]
             Temp = ic[1]

             #Loop until current temperature is <=30 degree Celsius
             while(Temp>stop_temp):

                 k1 = h*T_prime(t_current,Temp)
                 k2 = h*T_prime(t_current+h/2,Temp+1/2*k1)
                 k3 = h*T_prime(t_current+h/2,Temp+1/2*k2)
                 k4 = h*T_prime(t_current+h,Temp+1*k3)
```

```
        k =(k1+(2*k2)+(2*k3)+k4)

        Temp = (Temp + (k/6))   # update temperature using RK-4
        t_current += h          # increment time by step size

    #Return time at which temperature dropped to 30 degree Celsius
    return t_current



#STEP SIZE: 1
ic = np.array([0., start_temp])
h = 1.0
t_stp1 = R_K(ic, h)

print "Time approximation with step size ", h, " is: ", t_stp1, "seconds "

#STEP SIZE: 0.1
ic = np.array([0., start_temp])
h = 0.1
t_stp2 = R_K(ic, h)

print "Time approximation with step size ", h, " is: ", t_stp2, "seconds "

#STEP SIZE: 0.01
ic = np.array([0., start_temp])
h = 0.01
t_stp3 = R_K(ic, h)

print "Time approximation with step size ", h, " is: ", t_stp3, "seconds "
```

```
Time approximation with step size  1.0  is:  51.0 seconds
Time approximation with step size  0.1  is:  51.3 seconds     ✓
Time approximation with step size  0.01  is:  51.3 seconds
```

3.0.3 Verification using SymPy and application of initial conditions [10]

Use SymPy to find the general solution to the above problem and find C_1 using the initial conditions given. Using this particular solution, **show that the solution obtained using Runge-Kutta is correct.**

```
In [16]: # Code in SymPy here
         t = sp.Symbol('t')
         T=sp.Function('T')

         T_prime=sp.Derivative(T(t),t)
         eq=T_prime+0.05*(T(t)-20)
```

```
        sp.init_printing(use_latex=True)
        sp.dsolve(eq)
```

Out[16]:

$$T(t) = C_1 e^{-0.05t} + 20.0$$

Apply initial conditions and required temperature to solve for t here:
Now, we have, $T(0) = 150°C$
So,

$$\begin{aligned} & T(t) = C_1 e^{-0.05t} + 20.0 \\ \implies & T(0) = C_1 e^{-0.05*0} + 20.0 \\ \implies & 150.0 = C_1 + 20.0 \\ \implies & C_1 = 130.0 \end{aligned}$$

Let t_i be the time when Temperature drops to $30°C$,
So,

$$\begin{aligned} & 30.0 = 130.0 \cdot e^{-0.05t_i} + 20.0 \\ \implies & e^{-0.05t_i} = \frac{10}{130} \\ \implies & -0.05t = \ln(\frac{1}{13}) \\ \implies & t_i = 51.23 \end{aligned}$$

✓

80/80