Control System I
Python exercises summary

September 20, 2018

Summary

Recap of the proposed exercises. When you do not remember how to use a function and which arguments it takes use the command `help(function)`.

Exercise #0

Print all integers numbers from 0 to 20

- If a number is divisible by 3 print "Buzz" instead
- If a number is divisible by 3 print "Buzz" instead
- If it is divisible by both 3 and 5 print only the number.

All numbers must be printed on a new line. Save the output to a text file called `ex0_output.txt`.

Exercise #1

Write a program that accepts a sequence of whitespace separated words as input and prints the words after removing all duplicate words and sorting them alphanumerically. Suppose the following input is supplied to the program:

today is a beautiful day almost perfect today is beautiful did I say day ?

Hint: use `input("prompt string")` to get the input words list from terminal.

Exercise #2

Write a program that takes as a input a sentence and that then saves to a dictionary the number of digits and of characters contained in the sentence. Print the dictionary. e.g:

- **Input**: *Today is 17 September 2018.*
- **Output**: {`DIGITS: 6, CHARACTERS: 16`}

Hint: you can use the methods `isdigit()` and `isalpha()`

Exercise #3

Write a binary function which searches an item in a *sorted list*. The function should return the index of element to be searched in the list.

Binary search It works looking at the middle element of the list and takes the left or right half if the item is less or greater then the middle element respectively. The procedure is then repeated until the element is found.

Hint: `import math` for the `floor()` and `ceil()` functions.

Exercise #4

ODE problems are important in computational physics, so we will look at one more example: *the damped oscillator*. This problem is well described on the wiki page: `http://en.wikipedia.org/wiki/Damping`. The equation of motion for the damped oscillator is:

$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2 x = 0 \tag{1}$$

where x is the position of the oscillator, ω_0 is the frequency, and ζ is the *damping ratio*. To write this second-order ODE on standard form we introduce a 2-dimensional vector defined as follow:

$$\dot{x}_1 = x_2 \tag{2}$$

$$\dot{x}_2 = -2\zeta\omega_0 x_2 - \omega_0^2 x_1 \tag{3}$$

Then the ODE can be expressed in matrix form:

$$\begin{bmatrix}\dot{x}_1 \\ \dot{x}_2\end{bmatrix} = \underbrace{\begin{bmatrix}0 & 1 \\ \omega_0^2 & -2\zeta\omega_0\end{bmatrix}}_{:=A}\begin{bmatrix}x_1 \\ x_2\end{bmatrix} \tag{4}$$

In the implementation of this example we will add extra arguments to the RHS function for the ODE, rather than using global variables as we did in the Scipy notebook tutorial. As a consequence of the extra arguments to the RHS, we need to pass an keyword argument args to the odeint function:

```
odeint(dy, y0, t_c, args=( ... ))
```

where `args` takes all the arguments accepted by the system equation, namely `dy`.

An alternative to get an approximated solution to the differential equation is to use *forward Euler approximation*. In Euler method the differential equation is approximated as follows:

$$f(y,t) = \dot{y} \approx \frac{y(t+dt) - y(t)}{dt} \rightarrow y(t+dt) = y(t) + dtf(t,y) \tag{5}$$

where dt is the discretization time interval. The smaller the intervals the better the approximation is. In the linear case the equation becomes

$$y(t+dt) = (I + Adt)y(t) \tag{6}$$

where with I we denote the identity matrix whose dimensions are consistent with A.
Solve the ode in both cases using `odeint` from `Scipy` and `Numpy`. Compare and plot the solutions for the time interval `[0s, 10s]`. Test the following parameters (start with $dt = 0.01s$):

- $\omega_0 = 2\pi 1.0$
- $\zeta = [0.0, 0.2, 1.0, 5.0]$

What do you observe as the value of ζ changes ?
What do you observe as you increase the value of dt? (Especially consider the undumped case, aka $\zeta = 0.0$).
Hint: code skeleton:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint


# Discretized solution
def dyDt(y, dt, tend, zeta, w0):
    """
    Discrete time solution
    param y:    the initial condition
    param dt:   the discretization interval
    param tend: the final time
    param zeta: damping coefficient
    param w0:   natural frequency

    output sol: array (M, N) with M equal to the number of time
                steps and N=2, number of system states (in this
                                                        case x_1
                and x_2)
    """

    # TODO
    # Implement the forward Euler method

    return sol
```

```
# Continuous time solution
def dyCt(y, t, zeta, w0):
    """
    The right-hand side of the damped oscillator ODE
    """
    x1, x2 = y[0], y[1]

    # TODO
    # Compute dx_1 and dx_2

    return [dx_1, dx_2]


# initial state:
y0 = [1.0, 0.0]

# params dictionary
params = {"zeta": [0.0, 0.2, 1.0, 5.0], "w0": 2*np.pi*1.0,
          "label": ["Undamped", "Damped", "Critical", "Overdamped"]
                                                    }

# time coordinate to solve the ODE with odeint
t_c = np.linspace(0, 10, 1000)

# time coordinate to solve the ODE with Euler approximation
dt = 0.01
tend = 10

fig, ax = plt.subplots(4, 1, sharex=True)

# TODO
# Solve and plot for the 4 cases the odeint and dicrete time
# solutions

plt.show()
```