Python Exercises

Miletos Summer Camp

July 2018

1 Is Prime?

1.1 Question

Write a function that checks given number is prime or not.

Hints:

1. Go up to square root of number.

2 Replace Turkish Letters

Write a function that replaces Turkish letter into suitable English letters in a string. Do this for list of strings and write into a file. Select inputs as you wish.

Hints:

1. You can use *dictionary*.
2. You can use `map` function for list of strings.

3 Euler Method

Euler method is used to solve ordinary differential equations (ODE)s.

General first order differential equation form is

$$\frac{dy}{dt} = f(t, y), \qquad y(t_0) = y_0 \tag{1}$$

We want to approximate the solution to (1) near t_0. We only know the value of the solution and its derivative at initial point. We can get this by plugging the initial condition into $f(t, y)$ into the differential equation itself. So, the derivative at this point is.

$$\left.\frac{dy}{dt}\right|_{t=t_0} = f(t_0, y_0) \approx \frac{y - y_0}{t - t_0} \tag{2}$$

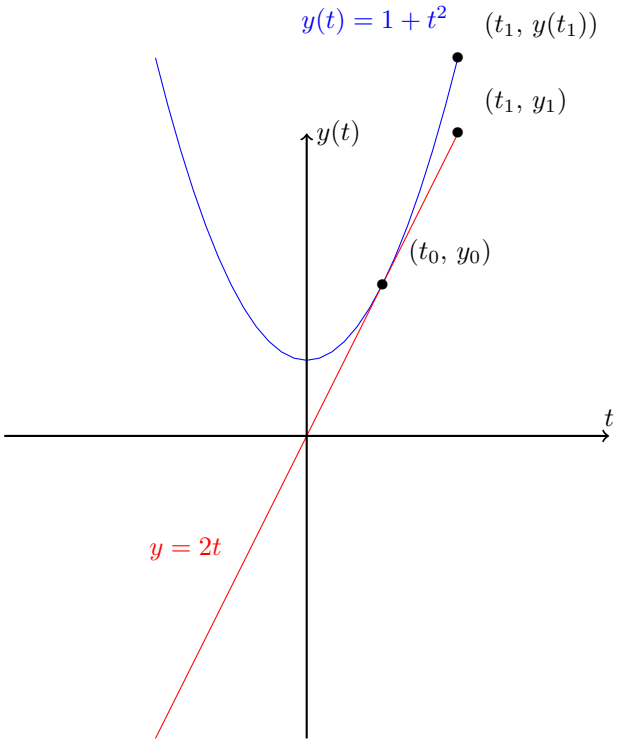

Figure 1: Sample approach for the plot of $f(x) = 1 + t^2$ with a tangent at $t = 1.0$.

If t_1 is close enough to t_0 then the point y_1 on the tangent line should be fairly close to the actual value of the solution at t_1. Finding y_1 is easy enough. All we need to do is plug t_1 in the equation for the tangent line.

$$y_1 = y_0 + f(t_0, y_0)(t_1 - t_0) \tag{3}$$

This y_1 is only an approximation. If we accept the error, we can continue build up lines similarly.

$$y_2 = y_1 + f(t_1, y_1)(t_2 - t_1) \tag{4}$$

$$y_3 = y_2 + f(t_2, y_2)(t_3 - t_2) \tag{5}$$

In general, starting from the initial point, (t_n, y_n) we have an approximation at (t_{n+1}, y_{n+1}) like

$$y_{n+1} = y_n + f(t_n, y_n)(t_{n+1} - t_n) \tag{6}$$

If we were to take a constant step size

$$t_{n+1} - t_n = h \tag{7}$$

Equation (6) would be,

$$y_{n+1} = y_n + h.f(t_n, y_n) \tag{8}$$

But of course this step size may not be constant, but adaptive.

We start at initial point (t_0, y_0) and repeatedly evaluate new points with selected or adaptively calculated step size. We continue until we reached the target point we want to compute (We will acquire $y(t_f)$ for target point t_f).

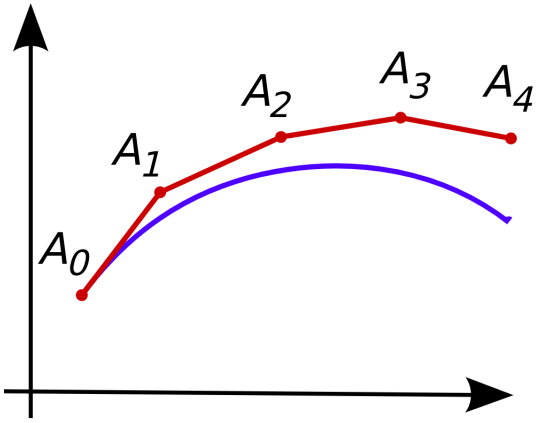

Figure 2: Euler approximation steps

Here is a pseudo-code for algorithm

define $f(t, y)$
input f, t_0, y_0, n(# of steps), t_f
compute step size h
for i from 1 to n
compute $f(t_0, y_0)$
compute corresponding y_1
compute new t_1
update t_0 and y_0
end
output y_0

3.1 Question a)

Write a function that takes function f, initial points t_0 and y_0, # of steps n and target point t_f and returns approximate y_f value.

3.2 Question b)

Test your function for following differential equations with given initial points at desired target points with different number of steps n. Compare your results to given analytic solutions. Show your numerical results and analytic results and their relative errors for different n' s.

$$i) \quad \frac{dy}{dt} = y + t \qquad y(0) = 0 \qquad \textbf{analytic solution:} (y = e^t - t - 1)$$

$$ii) \quad \frac{dy}{dt} = \sin t - \frac{y}{t} \qquad y(0) = 0 \qquad \textbf{analytic solution:} (y = \frac{\sin t}{t} - \cos t)$$

Hints:

1. Use only `math` library.
2. relative error = $\frac{|\text{predicted value} - \text{true value}|}{\text{true value}}$

4 Computing π with Monte Carlo Simulation

Monte Carlo methods are a broad class of computational algorithms based on repeated random sampling used for making numerical computations.

You will implement such an experiment to compute approximate value of π.

In Monte Carlo simulations two basic sampling methods exist. One of them is *direct sampling*. In the direct sampling method, we take samples independently for each step by choosing a random position in the space. Hence, if we use real random numbers, our new samples are independent of the previous ones, because we choose a new position randomly. By taking samples again and again, we sweep out the space and by using these samples, we can make calculations.

Other sampling method is *Markov-Chain sampling*. In the Markov-Chain sampling, our sample depends on the previous sample. Firstly, we start sampling from an initial position that is given or where the last simulation ends. Then, from the initial site, we move to another site on the space, in any direction and distance. This direction and distance again random, but, distance is limited to a value δ. By using this procedure, we visit other states and if the resulting state out of our space, then we reject the move. Hence, this limitation for the moves δ affects the rejection rate. If δ is very large, rejection rate shold be too high, this means that we generally do not move to another state; thus, our traveled path is small. Also, if δ is very small, acceptance rate becomes too high, this means that we generally move from state to another state; however, in this case our traveled path is, again, small and we cannot sweep out the space.

In this question you select random points with these two methods in space with circle and square where r is the radius of the circle and $2r$ is the length of square as seen below.

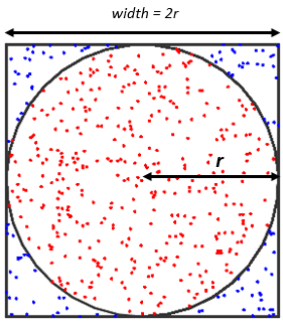

Figure 3: Collection of random selected points in sample space (tangential circle in a square)

4.1 Question a)

Use *direct sampling* to simulate experiment and compute approximate π value. Your simulations should take two parameters: *power* and *repetition*. Power represents number of samples. For example if power = 5, then number of samples should be 10^5. *Repetition* represents number of repetition of one simulation to take the average of the π value. For example run your simulation with *repetition* = 10 and take the average π for those 10 simulations. Compare your results to real π value for different *power* and *repetition* values.

4.2 Question b)

Use *Markov-chain sampling* to simulate experiment and compute approximate π value. Start from random point in the space and move by random δ_x and δ_y along each axis at each step. If you go out of space, reject that point and choose different random points. Evaluate π values and rejection rate by keeping step size constant and changing number of samples; and keeping number of samples constant and changing step size.

Hints:

1. Use only `math` and `random` libraries.
2. Use circle-to-square area ratio and solve for π.