Homework 1 –

NUMA01: Computational Programming with Python

Claus Führer

You should upload your solution for this homework **at the latest on**

Friday May 31, 2019

This assignment has 7 tasks.

All the functions must be *properly documented* and also tested. You may work and present in groups by two-three. Clearly, working in groups with more members during the process of understanding the problems and making *first* design steps towards a code is allowed and even encouraged. On the course assignment tab in the course page you can upload your code. Write your code in such a way, that when executed it produces the answers to *all* questions. Comment the code in such the way, that it will help you, when later presenting it in an oral presentation.

Quadrature

Theory

In this homework we will compute approximations to the integral

$$I = \int_a^b f(x)\,\mathrm{d}x. \tag{1}$$

One method for doing this is by using the *composite trapezoidal rule*, given by the formula

$$I_h = \frac{h}{2}(f(a) + f(b)) + h\sum_{i=1}^{n-1} f(x_i) \tag{2}$$

where $x_i = a + \frac{i}{n}(b-a)$, $h = \frac{b-a}{n}$.

Then $I_h \approx I$ and the approximation gets better the smaller h is, i.e. the more points we divide the interval into. (Compare with the definition of the Riemann integral.)

Task 1

Write a function `ctrapezoidal(f,a,b,n)` which implements the trapezoidal approximation (2). Test this function for different n and compare your result to the exact integral. (Choose a simple function f that you can integrate by hand, e.g. e^x. However, don't make it *too* simple.)

Task 2

Write a program that calls `ctrapezoidal(f,a,b,n)` for an increasing number of discretization points n in a loop. Stop the loop when the difference of two successive results is less than a given tolerance and return the final approximation.

Task 3

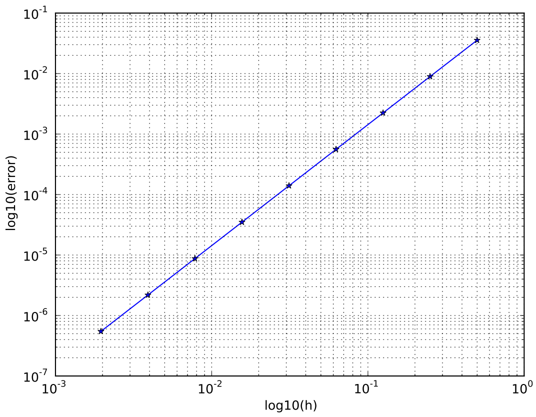
Write a function that makes an accuracy plot of the type depicted in the following figure. This plots the error $|I_h - I|$ against the step size h.

We make a loglog-plot, because we expect the error to be proportional to h^2 for small h, that is, error $= Ch^2$. Taking the logarithm of both sides yields

$$\log \text{error} = 2 \log h + \log C.$$

This means that we should see a straight line of slope 2 if everything is correct. You do not need to take all $n = 1, 2, 3, \ldots$, it is enough to take e.g. $n = 1, 2, 4, 8, \ldots$.

(See the command `loglog` for making figures in a double logarithmic scale and `grid` for turning on the grid.)



Amorizations and Interests

Theory

Assume someone wants to take a loan K_0 on January, 1 of a certain year – say Year 0. The interest (ränta, sv.) for this loan is $r\%$. Furthermore the person pays an annual amortization R at December, 31. The following formula can be used to calculate the remaining loan K_n after n years:

$$K_i = K_{i-1}\left(\frac{r}{100} + 1\right) - R \qquad i = 1, \ldots, n$$

Task 4

Write a function `sparbanken(K_0,r,n,R)` which computes the remaining loan after n years assuming that the initial loan was K_0 and the interest $r\%$. R denotes the yearly (constant) amortization. Write a clear and explanatory `docstring` to this function. Make sure that the input quantities are checked to be positive numbers, where K_0, R and the result should be rounded to full crowns if it was given with decimals (see Python's function **`round`**).

Task 5

Write another function which determines the first year when the loan is fully payed back. This function should call the function you wrote for the previous task. How you determine this year is up to you. If you "invented" some method for this be prepared to explain it.

Task 6

Write a function which determines an ammortization rate such that the loan is payed back after exactly n years. Also here, how you determine this rate is up to you. But be prepared to explain your approach.

Task 7

Make a plot for the client to demonstrate him or her how the loan develops over the years. This plot should contain at least three curves for different interests r or amortizations R.

Good luck!