LUND UNIVERSITY

Homework 1 – 2016-04-17

NUMA01: Computational Programming with Python

Claus Führer

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

2016-04-25

This assignment has 7 tasks.

All the functions must be *properly documented* and also tested. We recommend that you produce a report of your work with IPython-Notebook (see lecture on Monday April 18). You may work and present in groups by two. Use the upload link `http://www.maths.lth.se/na/courses/NUMA01/dl/` to upload your code. Upload one file only, either a *.py-file or an IpythonNotebook file. You will get then an email later from one of the teaching assistants with a presentation time slot.

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

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

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

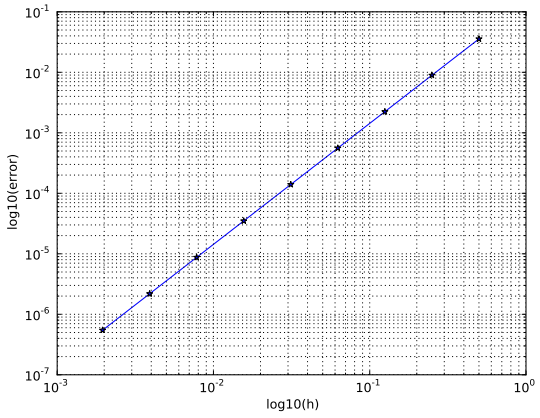
Write a function that makes an accuracy plot of the type depicted in the following figure. This plots the error $|I_h - I|$ against the step size h.

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.)



Interpolation

Theory

We consider a method for *interpolating* a sequence of points, that is, finding a polynomial P (of lowest degree) such that $P(x_i) = y_i$ for given points -often measurements- (x_i, y_i) . If $N+1$ points are given, there is a unique polynomial of degree N which passes through these points.

How to find the interpolation polynomial?
Let

$$P(x) = c_N x^N + c_{N-1} x^{N-1} + \cdots + c_1 x + c_0$$

be a polynomial of degree N which interpolates the $N+1$ points (x_i, y_i), i.e. $P(x_i) = y_i$. For its coefficients we get the equation system

$$P(x_i) = c_N x_i^N + c_{N-1} x_i^{N-1} + \cdots + c_1 x_i + c_0 = y_i, \quad i = 0, \ldots, N.$$

We write this on matrix form:

$$\begin{pmatrix} x_0^N & x_0^{N-1} & \cdots & x_0^1 & x_0^0 \\ x_1^N & x_1^{N-1} & \cdots & x_1^1 & x_1^0 \\ \vdots & & \ddots & & \vdots \\ x_N^N & x_N^{N-1} & \cdots & x_N^1 & x_N^0 \end{pmatrix} \begin{pmatrix} c_N \\ c_{N-1} \\ \vdots \\ c_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{pmatrix} \tag{3}$$

and denote the matrix V, the coefficient vector c and the right-hand side y, so that

$$Vc = y.$$

Task 4

Given a vector x of length $N+1$, write a Python function that constructs the matrix

$$\begin{pmatrix} x_0^N & x_0^{N-1} & \cdots & x_0^1 & x_0^0 \\ x_1^N & x_1^{N-1} & \cdots & x_1^1 & x_1^0 \\ \vdots & & \ddots & & \vdots \\ x_N^N & x_N^{N-1} & \cdots & x_N^1 & x_N^0 \end{pmatrix}$$

where the x_i are the components of the vector x (numbered from zero). Recall the function `column_stack` to stack vectors horizontally.

Task 5

Write a function `interpoly` that computes the coefficient vector c according to (3), given the vectors x and y. You can use `scipy.linalg.solve` to solve the equation system.

Task 6

Write a function `polyval`, which has c and z as input and which computes the polynomial

$$P(z) = \sum_{i=0}^{N} c_i z^i.$$

Task 7

Test these last three functions on the vectors

$$x = (0.0, 0.5, 1.0, 1.5, 2.0, 2.5)$$
$$y = (-2.0, 0.5, -2.0, 1.0, -0.5, 1.0)$$

by plotting the polynomial P over $[0, 3]$. Plot also the points (x_i, y_i) as small stars and make sure that the polynomial passes through these points.

Good luck!