Programmierpraktikum

Exercise Sheet #8

WS, 2012/2013

Numerical Integration

Write two functions that calculate the integral:

$$y_m = \int_a^b \cos^m(\alpha x)\mathrm{d}x$$

using

- the trapezoid rule (as explained the class);
- the rectangle rule (like the trapezoid, but assuming $f(x_i \leq x < x_{i+1}) = f(x_i)$);

Notice all variables should only be given as arguments to the function when called (this includes the number of sample points to be taken).

Use these functions to calculate the values of the integral for the arguments:

1. $a = 0; b = 3; m = 1; \alpha = 0.5$
2. $a = 1; b = 2; m = 2; \alpha = 2$

Calculate the integrals for increasing number of sample points (i.e. increasing number of intervals, as in the link below) http://itp.uni-frankfurt.de/~gros/Vorlesungen/ProgPrak/methods-evaluation.html#%283%29

Check that your calculation is correct: compare with analytical and/or tabulated results for a few cases.

How many sample points does each method need to converge in each case?

Convergence acceleration

http://itp.uni-frankfurt.de/~gros/Vorlesungen/ProgPrak/methods-evaluation.html#%286%29

Follow the derivation for the convergence acceleration seen in the class.

- How can we speed up an algorithm scaling as n^{-z}, for any $z \leq 1$?

Buffered Input/Output, IO streams as arguments in function calls

Write this exercise such that each point can be solved by giving each IO stream as an argument to a function.

IO operations are slow, sometimes *really* slow. To realize this, do the following:

- Write four programs, each one calculating the first million integers divisible by 7 and
  1. printing each number to the standard output;
  2. printing each number to a file;
  3. printing each number to a file, but using buffering
     (`[Optional]` try flushing the buffer at different periods to find out if there is an optimum);
  4. does not print anything, just calculates the values.

In each case, print the time that each program needs to complete. You can do this by using the unix command `time`, or by using the time measurement provided by Java, for instance retrieving `System.nanotime()` (from `java.lang`) before and after your code, and then calculating the difference.