Exercise n. II

October 10, 2019

1 Root-finding

The bisection method is a root-finding method that applies to any continuous functions for which two values of opposite sign are known:

- Algorithm: given a continuous function f, an interval $[a, b]$, and the function values $f(a)$ and $f(b)$. If the function values are of opposite sign, then there is at least one zero crossing within the interval. Each iteration performs these steps:
  1. calculate the midpoint $m = (a + b)/2$;
  2. calculate $f(m)$;
  3. if $f(m)$ is "sufficient small" then return m;
  4. otherwise, check the sign of $f(m)$ and replace either $(a, f(a))$ or $(b, f(b))$ with $(m, f(m))$ so that there is a zero crossing within the new interval and iterate.
- Implement the algorithm using the function $f(x) = 2x^3 - 4x + 1$.
- Hints: use functions for the evaluation of $f(x)$ and for the root-finding algorithm. Solution in *zero_function.c*

2 Machine Precision

The machine precision ϵ_m is defined as the smallest positive number that added to the unit does change its value stored in memory, i.e. $\epsilon_m + 1 = 1$. NOTE: It's not the smallest representable number!

For example the pseudocode could be:

$\epsilon = 1$
do until... (or do N times, with $N = ...$)
    $\epsilon = \epsilon/2$
    $one = 1 + \epsilon$
    write number of iterations and the value of ϵ.

Check the machine precision for floating point in single and in double precision and compare with the values *FLT_EPSILON* and *DBL_EPSILON* provided by <*float.h*>.

Hint: Solution in *epsilon.c*

3 Bad and good algorithms, truncation and round-off

A typical numerical problem is to calculate a function for a given value of a variable as the sum of a series. For instance:

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} + ...$$

- Write a program to calculate in single precision e^{-x} as the *sum* of the series above and compare the result with the *exp* function provided by the *C* math library, for instance evaluating the relative error as $|(sum - exp(-x))/exp(-x)|$;
- start from $x = 0$ up to $x \simeq 50$, with a step $\Delta \simeq 0.1$;
- using the *factorial function* make some tests fixing the number of terms of the series. Then, try to figure out how to avoid the factorial in order to increase the number of terms. Which program works better and why?
- consider the best code. Change the code in order to calculate $e^{-x} = 1/e^x$. It is better? Why?
- discuss the results in double precision.

Hint: Solution in *exp_bad.c*