ITU Computer and Informatics Faculty
BLG 202E Numerical Methods in Computer Engineering
2020 - 2021 Spring
Homework 1

Handed out: 24.03.2021
Due: 07.04.2021 23:59

Notes:

- Prepare a report for this homework in PDF format. You can use word, latex or you can use your handwriting. The handwritten parts of the solutions must be presented on a paper legibly and scanned clearly.
- Only **one** page should be used for each answer.
- The written Python codes should be uploaded separately.
- Please do not forget to write your name and number at the top of each file you submitted.
- Please submit your report through Ninova e-Learning System. Another way of submission will not be accepted. Also, the late submissions will **NOT** be accepted.
- In the case of cheating and plagiarism, strong **disciplinary action will be taken**.
- For any questions about the Homework 1, contact T.A. Beyza Eken (beyzaeken@itu.edu.tr).

Questions:

1. The function $f_1(x_0, h) = \sin(x_0 + h) - \sin(x_0)$ can be transformed into another form, $f_2(x_0, h)$ using the trigonometric formula given below:

$$\sin(\phi) - \sin(\psi) = 2\cos\left(\frac{\phi + \psi}{2}\right)\sin\left(\frac{\phi - \psi}{2}\right)$$

   Thus, f_1 and f_2 have the same values, in exact arithmetic, for any given argument values x_0 and h.

   a. Derive $f_2(x_0, h)$.

   b. Suggest a formula that avoids cancellation errors for computing the approximation $(f(x_0 + h) - f(x_0))/h$ to the derivative of $f(x) = \sin(x)$ at $x = x_0$. Write a Python program that implements your formula and computes an approximation of $f'(1.2)$, for $h =$ 1e-20, 1e-19, …, 1.

2. Consider the linear system

$$\begin{pmatrix} a & b \\ b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

with $a, b > 0$.

a. If $a \approx b$, what is the numerical difficulty in solving this linear system?

b. Suggest a numerically stable formula for computing $z = x + y$ given a and b.

c. Determine whether the following statement is true or false, and explain why:

"When $a \approx b$, the problem of solving the linear system is ill-conditioned but the problem of computing $x + y$ is not ill-conditioned."

3. With exact rounding, we know that each elementary operation has a relative error which is bounded in terms of the rounding unit η; e.g., for two floating point numbers x and y, $fl(x + y) = (x + y)(1 + \epsilon), |\epsilon| \leq \eta$. But is this true also for elementary functions such as sin, ln, and exponentiation?

   Consider exponentiation, which is performed according to the formula below.

$$x^y = e^{y ln(x)} \; (assuming \; x > 0)$$

   Estimate the relative error in calculating x^y in floating point, assuming $fl(lnz) = (lnz)(1 + \epsilon), |\epsilon| \leq \eta$, and that everything else is exact. Show that the sort of bound we have for elementary operations and for ln does not hold for exponentiation when x^y is very large.

4. You are required by a computer manufacturer to write a library function for a given floating point system to find the cube root $y^{1/3}$ of any given positive number y. Any such relevant floating-point number can be represented as $y = a * 2^e$, where a normalized fraction $(0.5 \leq a < 1)$ and e is an integer exponent. This library function must be very efficient and it should always work. For efficiency purposes it makes sense to store some useful constants ahead of computation time, e.g., the constants $2^{1/3}, 2/3$ and $a/3$, and should these prove useful.

   a. Show how $y^{1/3}$ can be obtained, once $a^{1/3}$ has been calculated for the corresponding fraction, in at most five additional flops.

   b. Derive the corresponding Newton iteration. What is the flop count per iteration?

   c. How would you choose an initial approximation? Roughly how many iterations are needed? (The machine rounding unit is 2^{-52}.)