Homework Assignment 1

Kovalev Vyacheslav

(1). Check if the following equality holds using MATLAB:
0.1 + 0.2 == 0.3
Try to represent distinct numbers using fprintf('%.20e',x) and explain the result.

Machinery 0.1 in binary system is not the same as 0.1 in the decimal system. When we try to store 0.1 in a machine, it stores binary numbers the closest to decimal 0.1 with machine precision. The same to 0.2. So, when we add 0.1 to 0.2 it shifts the mantissa bits of 0.1 (according to the exponent) and sums it to the mantissa of 0.2, but some bits are out of available mantissa bits and system rounds. If we subtract 0.1 + 0.2 and 0.3 we can reveal this rounding error (5.5511e-17) – that is why (0.1 + 0.2 == 0.3) == false.

Using fprintf('%.20e',x) we can represent this rounding error:

Binary 0.3 is 2.99999999999999988898e-01
Binary 0.1 + 0.2 is 3.00000000000000044409e-01

(2). Check the associativity of summation:
(0.1 + 0.2) + 0.3 == 0.1 + (0.2 + 0.3)
Find the exponents of these numbers in decimal format for 32-bit float according to IEEE Standard with the help of an online converter (e.g. https://baseconvert.com/ieee-754-floating-point). Explain the loss of accuracy during floating-point addition on a binary level. Is the loss of accuracy always guaranteed?

$0.1 \approx 1.m_1 * 2^{01111011_2 - 127} = 1.m_1 * 2^{123-127} = 1.m_1 * 2^{-4}$
$0.2 = 0.1 * 2 \approx 1.m_1 * 2^{-3}$
$0.3 \approx 1.m_2 * 2^{-2}$

$(0.1 + 0.2) + 0.3 == 0.1 + (0.2 + 0.3)$ – false because as in first task:
$(0.1 + 0.2) = 1.m_1 * 2^{-4} + 1.m_1 * 2^{-3} = 2^{-3}(0.1m_1 + 1.m_1) = 1.M * 2^{-2}$(where extra digits of m_1 supposed to be 0)
But M contain more digits than mantissa can contain, that is why it rounds M. So, sum all up: $(0.1 + 0.2) = 1.M * 2^{-2} = 1.m_2 * 2^{-2} + roundErr_1$

$$(0.1 + 0.2) + 0.3 = 1.m_2 * 2^{-2} + roundErr_1 + 1.m_2 * 2^{-2}$$
$$= 1.m_2 * 2^{-1} + roundErr_1$$

Similar

$$0.1 + (0.2 + 0.3) = 1.m_1 * 2^{-4} + (1.m_1 * 2^{-3} + 1.m_2 * 2^{-2})$$
$$= 1.m_2 * 2^{-1} + roundErr_2$$

There are different round errors.
Is the loss of accuracy always guaranteed? – No.
For instance, when we don't need to perform rounding or rounding always correct.
(0.28888 + 0.29131 == 0.58019) – true (the same exponent, rounding correct)
Or when rounding always correct:
(16 + 4 ==20) - true (last digits are 0, therefore rounding correct, but different exponent)

(3). Check if the following equalities hold using MATLAB:
(2ˆ53 + 1) −2 ˆ 53 == 1
(2ˆ53 + 2) −2 ˆ 53 == 2
Explain the result.

(2ˆ53 + 1) −2 ˆ 53 == 1 – false
(2ˆ53 + 2) −2 ˆ 53 == 2 – true

$$2^{53} + 1 = 1 * 2^{53} + 0.0 \ldots 01 * 2^{53} = 1.0 \ldots 01 * 2^{53}$$

The mantissa of this number contains all zero and 1 on the 53rd position, but in 64-bit IEEE 754 system expects 52 bit on mantissa, therefore last digits will be rounding to the nearest integer and last digits will be cut. consequently only 2^{53} will remain. So, we lose 1.
$2^{53} + 1 = 2^{53}$ - true

Similar for $2^{53} + 2$, but the mantissa of this number contains all zero and 1 on the 52nd position. => 2 corresponds to 1 bit, the last bit of mantissa, therefore it will remain.
$2^{53} + 2 = 2^{53}$ - false