Study Questions Homework 1 Foundations of Computational Math 2 Spring 2025

Questions 4, 7 and 9 are the more advanced questions while the others are at a basic level.

Problem 1.1

The IEEE standard defines the set of normalized floating point numbers with base 2 as

$$f = \pm\, 2^e \times (1.b_1 b_2 \dots b_n)_2 = \pm\, 2^e \times \left(1 + \sum_{k=1}^{n} b_k 2^{-k}\right)$$

$$b_k \in \{0,\ 1\},\ 1 \le k \le n,\ \ L_{stand} \le e \le U_{stand}$$

with

- Single precision
$$n = 23,\ \ L_{stand} = -126,\ \ U_{stand} = 127$$
- Double precision
$$n = 52,\ \ L_{stand} = -1022,\ \ U_{stand} = 1023.$$

1. For both the single and double precision binary standard representations, find the
   - The number of normalized mantissas.
   - The number of exponents.
   - The number of elements in the set of IEEE normalized positive numbers.
   - f_{min} the smallest normalized positive floating point number.
   - f_{max} the largest normalized positive floating point number.
   - Machine epsilon, ϵ_M, defined as the distance from 1 to next positive normalized floating point number.
   - The absolute difference between normalized positive floating point numbers.
2. Recall, that we use a standard parameterized representation of normalized floating point numbers $\mathcal{F}(\beta, t, L, U)$:
$$f = \pm\, \beta^e \times (0.d_1 d_2 \dots d_t)_\beta = \pm\, \beta^e \times \left(\sum_{k=1}^{t} d_k \beta^{-k}\right)$$
$$d_k \in \{0,\ 1, \dots, \beta - 1\},\ 2 \le k \le n, \quad d_1 \in \{1, \dots, \beta - 1\},\ \ L \le e \le U.$$

Determine the value of the parameters β, t, L, and U so that $\mathcal{F}(\beta, t, L, U)$ is the set of normalized floating point numbers defined by the single and double precision IEEE binary standard.

3. Verify that the quantities you computed in part (1) of this question match the values of the expressions for $\mathcal{F}(\beta, t, L, U)$ given in the notes.

Problem 1.2

Suppose the n-bit 2's complement representation is used to encode a range of integers, $-2^{n-1} \leq x \leq 2^{n-1} - 1$.

1.2.a. If $x \geq 0$ then $-x$ is represented by bit pattern obtained by complementing all of the bits in the binary encoding of x, adding 1 and ignoring all bits in the result beyond the n-th place, i.e., the bit with weight 2^{n-1}. This procedure is also used when $x < 0$ to recover the encoding of $-x \geq 0$. What is the relationship between the binary encoding of $-2^{n-1} \leq x \leq 2^{n-1} - 1$ and the binary encoding of $-x$ in terms of the number of bits n?

1.2.b. Show that simple addition modulo 2^n on the encoded patterns is identical to integer addition (subtraction) for $-2^{n-1} \leq x,\ y \leq 2^{n-1} - 1$. You may ignore results that are out of range, i.e., overflow.

1.2.c. Show how overflow in addition (subtraction) can be detected efficiently.

1.2.d. Multiplying an unsigned binary number by 2 or 1/2 corresponds to shifting the binary representation left and right respectively (a so-called logical shift). Show how multiplying signed integers encoded via 2's complement representation by 2 or 1/2 can be done via a shifting operation (an arithmetic shift).

Problem 1.3

1.3.a. Suppose $x \in \mathbb{R}$ and $y \in \mathbb{R}$ with $x < y$. Is it always true that $fl(x) < fl(y)$ in any standard model floating point system?

1.3.b. Suppose x, y and z are floating point numbers in a standard model floating point arithmetic system. Is floating point arithmetic associative, i.e., is it true that

$$(x \boxed{op} (y \boxed{op} z)) = ((x \boxed{op} y) \boxed{op} z) \ ?$$

1.3.c. Is floating point arithmetic distributive, i.e., is it true that

$$fl(fl(x + z) * y) = fl(fl(fl(x * y) + fl(y * z)))?$$

1.3.d. Suppose x and y are two floating point numbers in a system $\mathcal{F}(\beta, t, L, U)$ with opposite signs. How close do x and y have to be in magnitude in order for the result of the floating point computation

$$(x \boxed{+} y)$$

to be exact?

Problem 1.4

Most recent machines use a binary base,$\beta = 2$, but the number of bits, t, may vary in a floating point system. The following algorithm is an attempt to determine t experimentally.

$x = 1.5$, $u = 1.0$,$t = 0$, $\alpha = 1.0$
while $x > \alpha$
    $u = u/2$
    $x = \alpha + u$
    $t = t + 1$
end

1.4.a. Assume that the floating point system has $\beta = 2$ and uses a hidden bit normalization. Does this algorithm find t? Assuming you implement the code in a high level language and ther operations are done in floating point arithmetic, does the method of rounding affect your answer?

1.4.b. Apply the algorithm to a machine that uses $\beta = 2$ in single precision and double precision. Do your observations from the output of the code agree with the IEEE floating point standard for single and double precision floating point numbers? (Note that this is not a programming assignment and you are not required to submit a solution. You are however encouraged to implement this algorithm in any language you find appropriate in order to answer this question.)

Problem 1.5

Consider the function

$$f(x) = \frac{1.01 + x}{1.01 - x}$$

1.5.a. Find the absolute condition number for $f(x)$.

1.5.b. Find the relative condition number for $f(x)$.

1.5.c. Evaluate the condition numbers around $x = 1$.

1.5.d. Check the predictions of the condition numbers by examining the relative error and the absolute error

$$err_{rel} = \frac{|f(x_1) - f(x_0)|}{|f(x_0)|}$$
$$err_{abs} = |f(x_1) - f(x_0)|$$

with $x_0 = 1$, $x_1 = x_0(1 + \delta)$ and δ small.

Problem 1.6

Let $f(\xi_1,\ \xi_2, \ldots, \xi_k)$ be a function of k real parameters ξ_i, $1 \leq i \leq k$. Recall, the relative condition number of f with respect to ξ_1 can be expressed

$$\kappa_{rel} = \max(1, c(\xi_1, \xi_2, \ldots, \xi_k))$$

where $0 \leq c(\xi_1, \xi_2, \ldots, \xi_k)$ is a value that indicates the sensitivity of f to small relative perturbations to ξ_1 as a function of the parameters ξ_i, $1 \leq i \leq k$. If $c(\xi_1, \xi_2, \ldots, \xi_k) \leq 1$ then f is considered well-conditioned. Additionally, however, when $c < 1$ its value gives important information. The smaller c is the less sensitive f is to a relative perturbations in ξ_1.

Let $n \geq 2$ be an integer and $\beta > 0$. Consider the polynomial equation

$$p(x) = x^n + \beta x - 1 = 0.$$

1.6.a. Show that the equation has exactly one positive root $\rho(\beta)$ such that $0 < \rho < 1$ for any $\beta > 0$.

1.6.b. Derive a formula for $c(\beta, n)$ that indicates the sensitivity of $\rho(\beta)$ to small relative perurturbations to β.

1.6.c. Derive a lower and upper bound on $c(\beta, n)$.

1.6.d. Comment on the conditioning of $\rho(\beta)$ with respect to β and in particular for $n = 2$ and the conditioning as $\beta \to +\infty$.

Problem 1.7

The evaluation of

$$f(x) = x\left(\sqrt{x+1} - \sqrt{x}\right)$$

encounters cancellation for $x \gg 0$.

Rewrite the formula for $f(x)$ to give an algorithm for its evaluation that avoids cancellation.

Problem 1.8

For this problem assume that the floating point system uses $\beta = 10$ and $t = 3$. The associated floating point arithmetic is such that $x \boxed{op} y = fl(x\ op\ y)$.

Let x and y be two floating point numbers with $x < y$ and consider computing their average $\alpha = (x + y)/2$.

Consider three algorithms for computing α. The parentheses indicate the order of the floating point operations.

- $\alpha_1 = ((x + y)/2.0)$
- $\alpha_2 = ((x/2.0) + (y/2.0))$
- $\alpha_3 = (x + ((y - x)/2.0))$

For the floating point values $x = 5.01$ and $y = 5.02$:

1.8.a. Evaluate α_1, α_2, and α_3 in the specified floating point system.

1.8.b. Explain the results.

1.8.c. Some algorithms produce a series of intervals by splitting an interval (x, y) into intervals (x, α) and (α, y) and choosing to process one of these two smaller intervals further in the next step of the algorithm. Could the behavior observed for the three average computations cause difficulties for such an algorithm?

Problem 1.9

Consider the summation $\sigma = \sum_{i=1}^{n} \xi_i$ using the following "binary fan-in tree" algorithm described below for $n = 8$ but which clearly generalizes easily to $n = 2^k$:

$$\sigma = \{[(\xi_0 + \xi_1) + (\xi_2 + \xi_3)] + [(\xi_4 + \xi_5) + (\xi_6 + \xi_7)]\}$$

or equivalently

$$\sigma_j^{(0)} = \xi_j,\ 0 \le j \le 3$$

$$\sigma_0^{(1)} = \xi_0 + \xi_1, \quad \sigma_1^{(1)} = \xi_2 + \xi_3, \quad \sigma_2^{(1)} = \xi_4 + \xi_5, \quad \sigma_3^{(1)} = \xi_6 + \xi_7$$

$$\sigma_0^{(2)} = \sigma_0^{(1)} + \sigma_1^{(1)}, \qquad \sigma_1^{(2)} = \sigma_2^{(1)} + \sigma_3^{(1)}$$

$$\sigma = \sigma_0^{(3)} = \sigma_0^{(2)} + \sigma_1^{(2)}$$

In general, there will be $k = \log n$ levels and $\sigma = \sigma_0^{(k)}$. Level i has 2^{k-i} values of $\sigma_j^{(i)}$, each of which corresponds to a sum

$$\sigma_j^{(i)} = \xi_{2^i j} + \ldots + \xi_{2^i(j+1)-1}.$$

The algorithm is easily adaptable to n that are not powers of 2.

1.9.a. Derive an expression for the absolute forward error of the method for a fixed $n = 8$ or $n = 16$ and then generalize to $n = 2^k$.

1.9.b. Derive an expression for the absolute backward error of the method for $n = 8$ or $n = 16$ then generalize to $n = 2^k$.

1.9.c. Bound the errors and discuss stability relative to the simple sequential summation algorithm given by, for $n = 8$ but easily generalizable to any n,

$$\sigma = (((((((\xi_1 + \xi_2) + \xi_3) + \xi_4) + \xi_5) + \xi_6) + \xi_7) + \xi_8)$$

or equivalently

$$\sigma = \xi_1$$

$$\sigma \leftarrow \sigma + \xi_i, \quad i = 2, \ldots, 8$$