Foundations of Computational Math 1 Fall 2022

Graded Homework 1

Programming Exercise

1.1 Floating point system and addition

Consider a floating point system, $\mathcal{F}(t, \beta, e_{min}, e_{max})$, that stores t digits as an integer in base β with exponent lower bound e_{min} and upper bound e_{max}. The pair of integers (m, e) represents the floating point number

$$f = \pm m \times \beta^e = \pm(d_1 \times \beta^{t-1} + d_2 \times \beta^{t-2} + \cdots + d_t \times \beta^0) \times \beta^e$$

where $m \in \mathbb{Z}$, $0 \leq d_i < \beta$ are digits in m. Uniqueness of representation is enforced by the normalization of $d_1 \neq 0$. Note that this formula might be different from the definition in the class notes in order to allow the use of integer arithmetic in the routines below.

The following pseudocode routines implement translation, rounding and addition, $\boxplus$ in the floating point system $\mathcal{F}(t, 10, e_{min}, e_{max})$. Note that in the routines, the symbol $\div$ represents integer division $s \div b = \lfloor a/b \rfloor$, e.g., $7 \div 2 = 3$. The routines have the following functions:

Routine 1: Round-to-nearest

Routine 2: Translate a real number $x \in \mathbb{R}$ into a floating point number $fl(x)$, using chopping or round-to-nearest.

Routine 3: Floating point sum of $f_1, f_2 \in \mathcal{F}(t, 10, e_{min}, e_{max})$ using chopping or round-to-nearest.

Here are some important features of these routines.

- Using rounding strategy in routine 2: round to the nearest. A flag, *RoundtoNearest*, indicates in the translate and addition routines that round-to-nearest should be used.
- Default rounding strategy:

$$u = \begin{cases} u_c = \beta^{1-t} & \text{for round-toward-0 (chopping)} \\ u_n = \dfrac{\beta^{1-t}}{2} & \text{for round to nearest} \end{cases}$$

  u depends on your choice of programming language.
- The smallest normalized floating point number is $10^{e_{min}+t-1}$.

- Overflow and underflow detection is not included in the routines and should not be considered in the later tasks.

Tasks

1 Implementation

Implement routines 1, 2, and 3 with $t = 5, e_{min} = -9$ and $e_{max} = 0$.

2 Correctness test

1. Using round to nearest, empirically validate that the error of your translate routines satisfies $fl(x) = x(1+\delta), |\delta| < u_n$ where $|x| > 10^{e_{min}+t-1}$, i.e.

$$|\frac{fl(x) - x}{x}| = |\delta| \leq u_n$$

2. If $\boxplus$ satisfies the model for round-to-nearest,

$$\forall f_1, f_2 \in \mathcal{F}, f_1 \boxplus f_2 = (f_1 + f_2)(1+\delta), |\delta| < u_n,$$

then we must have the relative error bound

$$\frac{|(f_1 + f_2) - (f_1 \boxplus f_2)|}{|f_1 + f_2|} = |\delta| \leq u_n \tag{1}$$

Empirically validate the bound (1).

To empirically validate a numerical statement, you should run a large number of experiments with randomly generated data and record the results. Discuss the statistics of your results using means, variances and histograms. To verify an error bound, it is a good idea to use the histogram of the actual error achieved. You can also make key points about your observations by examining and explaining outliers in the results and the results of particular experiments designed to illustrate the key points.

Note that the exact real operations $+, -, \setminus$ are involved in the above discussion. Your discussions can evaluate them analytically, i.e., symbolically, when possible. In computational experiments, they can be approximated by the IEEE double precision arithmetic since it is suitably more accurate than the 5-digit decimal arithmetic you are analyzing.

3 Accumulation

Given a set of floating point numbers $\xi_{1:n} = (\xi_1, \cdots, \xi_n), \xi_k \in \mathcal{F}$, denote the sum by $s_n = \sum_{k=1}^{n} \xi_k$. Implement the routine 4 to compute $\hat{s}_n$, the floating point version of s_n.

For $\boxplus$, we have the following expression

$$\begin{aligned}
\hat{s}_2 &= s_2(1+\delta_2) \\
\hat{s}_3 &= (\hat{s}_2 + \xi_3)(1+\delta_3) \\
&= (s_2(1+\delta_2)+\xi_3)(1+\delta_3) \\
&= s_3 + s_2\delta_2 + s_3\delta_3 + O(u^2) \\
&\cdots \\
\hat{s}_n &= s_n + \sum_{k=2}^{n} s_k\delta_k + O(u^2)
\end{aligned}$$

1. For round to nearest,

$$\begin{aligned}
|\hat{s}_n - s_n| &\approx \left| \sum_{k=2}^{n} s_k\delta_k \right| \\
&\leq \sum_{k=2}^{n} |s_k| u_n \\
&\leq \sum_{k=2}^{n} \|\xi_{1:k}\|_1 u_n \\
&\leq \sum \|\xi_{1:n}\|_1 u_n = (n-1)\|\xi_{1:n}\|_1 u_n
\end{aligned} \tag{2}$$

   where $\xi_{1:k}$ denote the first k entries of the set $\xi_{1:n}$ and $\|\cdot\|_1$ denotes the vector 1-norm. Empirically validate the bound (2). Is this bound tight with respect to the actual error?

2. **Conditioning analysis:** The relative condition number of summing a set of real numbers $x_i, i = 1, \cdots, n$ is given by

$$\kappa_{rel} = \sup_{p_{1:n}\to 0} \frac{|\sum_{i=1}^{n} x_i - \sum_{i=1}^{n}(x_i + p_i)|}{|\sum_{i=1}^{n} x_i|} \frac{\|x_{1:n}\|_1}{\|p_{1:n}\|_1} \tag{3}$$

   where $p_{1:n}$ are perturbations on the data.
   Then (3) around u level can be approximated by randomly generating $|p_i| \leq |x_i u|$ and compute

$$c_{rel}^{p_{1:n}} = \frac{|\sum_{i=1}^{n} x_i - \sum_{i=1}^{n}(x_i + p_i)|}{|\sum_{i=1}^{n} x_i|} \frac{\|x_{1:n}\|_1}{\|p_{1:n}\|_1} \tag{4}$$

   Repeat (4) multiple times with random $p_{1:n}$ and approximate (3) with the maximum

$$\kappa_{rel} \approx \max_{p_{1:n}} c_{rel}^{p_{1:n}}.$$

   Randomly generate a set of set of number $x_i, i = 1, \cdots, 100$ in IEEE double precision. Estimate the condition number of the sum $s_{100}^{x_{1:100}}$ at u level with IEEE double precision arithmetic.

   Then consider the exact sum $s_{100}^{\xi_{1:100}}$ of $\xi_i = fl(x_i), i = 1, \cdots, 100$. Is the relative error between two exact sums consistent with your approximated condition number?

Submission of Results

Expected results comprise:

- A document describing your solutions as prescribed in the notes on writing up a programming solution posted on the Canvas.
- The source code, makefiles, and instructions on how to compile and execute your code including the Math Department machine used, if applicable.
- Code documentation should be included in each routine. (You don't need to paste your code in the writing report).
- All text files that do not contain code or makefiles must be PDF files. **Do not send Microsoft word files of any type.**

These results should be submitted by 11:59 PM on the due date. Submission of results is to be done via Canvas.

Algorithm 1: Round to nearest

Data: $remainder$, m ,e
Result: rounded number=(m, e)
if $remainder \geq 5$ **then**
    $m \leftarrow m + 1$;
    if $m = 10^t$ **then**
        $m \leftarrow m \div 10$;
        $e \leftarrow e + 1$;
    end
end

Algorithm 2: Translate

Data: $x \in \mathbb{R}$, RoundtoNearest
Result: $fl(x) = (m, e)$
Initialization: $s \leftarrow \text{sign}(x), a \leftarrow |x|, m, e, remainder$;
if $a > 10^t$ **then**
    $m \leftarrow \lfloor a \rfloor$;
    while $m \geq 10^t$ **do**
        $remainder \leftarrow m \bmod 10$;
        $m \leftarrow m \div 10$;
        $e \leftarrow e + 1$;
    end
    if *RoundtoNearest* **then**
        Call rounding routine 1;
    end
    $m \leftarrow s \times m$;
    Return $fl(x) = (m, e)$;
else
    while $a < 10^{t-1}$ **do**
        $a \leftarrow 10 \times a$;
        $e \leftarrow e - 1$;
    end
    $reminder \leftarrow \lfloor 10 \times (a - \lfloor a \rfloor) \rfloor$;
    $m \leftarrow \lfloor a \rfloor$;
    if *RoundtoNearest* **then**
        Call rounding routine 1;
    end
    $m \leftarrow s \times \lfloor a \rfloor$;
    Return $fl(x) = (m, e)$;
end

Algorithm 3: Addition

Data: $x = (m_x, e_x), y = (m_y, e_y) \in \mathcal{F}, RoundtoNearest$
Result: $x \boxplus y = (m, e)$
Initialization: Long integers a, b, c, integers $m, e, s, shift, remainder$;
$shift = e_x - e_y$;
if $shift > 0$ **then**
  $a \leftarrow m_x \times 10^{shift}$;
  $b \leftarrow m_y$;
  $e \leftarrow e_y$;
else
  $a \leftarrow m_y \times 10^{-shift}$;
  $b \leftarrow m_x$;
  $e \leftarrow e_x$;
end
$c \leftarrow a + b$;
$s \leftarrow \text{sign}(c)$;
$c \leftarrow |c|$;
if $c \geq 10^t$ **then**
  while $c \geq 10^t$ **do**
    $remainder \leftarrow c \bmod 10$;
    $c \leftarrow c \div 10$;
    $e \leftarrow e + 1$;
  end
else
  while $c < 10^{t-1}$ **do**
    $c \leftarrow c \times 10$;
    $e \leftarrow e - 1$;
  end
end
Cast long integer c to integer m, $m \leftarrow c$;
if *RoundtoNearest* **then**
  Call rounding routine 1;
end
$m \leftarrow s \times m$;
Return $x \boxplus y = (m, e)$;

Algorithm 4: Accumulation

Data: $x_k \in \mathcal{F},\ k = 1, \cdots, n$
Result: $\hat{s}_n$
Initialization: $s \leftarrow x_1$, $k \leftarrow 1$;
while $k < n$ **do**
  $s \leftarrow s \boxplus x_{k+1}$;
  $k \leftarrow k + 1$;
end
Return s;