Math 5630/6630 **Fall 2024**

Homework 1

Tags: Floating Point Arithmetics *Due Date: 09/17/2024 11:59PM CST*

1 Homework Problems

This part of the homework assignment should be submitted via Canvas. You can scan the answers into PDF files, or typeset them in Word or LaTeX, then convert/compile them into PDF files.

Problem 1.1. *Given a floating point system* $(\beta, t, L, U) = (10, 4, -3, 3)$*, where* β *is the base,* t *is the number of significant digits,* L *and* U *are the lower and upper bound of the exponent respectively.*

i. What are the floating point representations of the real numbers $x = 4.312809$ *and* $y = 0.4312809$ *if the usual rounding strategy is used? (note that the first digit* $d_0 \neq 0$*)*

ii. What is the rounding unit η *of this floating point system?*

iii. Which of the following four numbers will cause the problem of underflow or overflow in this floating point system?

(A) 0.0051; (B) 0.00000001; (C) 12000; (D) -12000.

Problem 1.2. *The function* $f_1(\alpha, h) = \sin(\alpha + h) - \sin(\alpha)$ *can be transformed into another form,* $f_2(\alpha, h)$ *using the following formula*

$$\sin(A) - \sin(B) = 2\cos\left(\frac{A+B}{2}\right)\sin\left(\frac{A-B}{2}\right).$$

Thus f_1 *and* f_2 *have the same values under exact arithmetic. In the following, we study which formula is better for calculating the approximation for* $f'(\alpha)$ *by*

$$\frac{f(\alpha + h) - f(\alpha)}{h}, \quad f(x) = \sin(x).$$

i. *Derive* $f_2(\alpha, h)$.

ii. *Suppose the evaluations of functions* ***cosine*** *and* ***sine*** *do not suffer from rounding errors (but all basic operations still do). Suggest a formula that avoids cancellation errors. Briefly explain why you made this suggestion.*

Problem 1.3. *Suppose a machine with a floating point system* $(\beta, t, L, U) = (10, 8, -50, 50)$ *is used to calculate the roots of the quadratic equation*

$$ax^2 + bx + c = 0,$$

where a, b, c *are given, real coefficients (in the floating point system). For each of the following, state the numerical difficulties that arise if one uses the standard formula*

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

for computing the roots. Explain how to overcome these difficulties (when possible).

i. $a = 1$; $b = -10^5$; $c = 1$.

ii. $a = 6 \cdot 10^{30}$; $b = 5 \cdot 10^{30}$; $c = -4 \cdot 10^{30}$.

iii. $a = 10^{-30}$; $b = -10^{30}$, $c = 10^{30}$.

Problem 1.4. *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 $a, b \in \mathbb{R}_+$.

i. Find the solution (x, y) *in terms of* a *and* b.

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

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

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

"When $a \approx b$, *solving the linear system for* x *and* y *is ill-conditioned* [1] *but the problem of computing* $x + y$ *is NOT ill-conditioned."*

[1] In this context, ill-conditioned means if you solve the same linear system with right-hand side as $(1 + \varepsilon_1, \varepsilon_2)^T$ that $|\varepsilon_1|, |\varepsilon_2|$ are tiny, the error in your desired quantity is huge.

1.1 Extra Problems for MATH 6630

Problem 1.5. *We study the rounding errors of summation* $S_n = \sum_{j=1}^n x_j$, *where* $x_j \in \mathbb{R}$.

i. Show the rounding error of S_n *using naive summation, that is,*

$$S_j = S_{j-1} \boxed{+} fl(x_j), \quad S_0 = 0,$$

is bounded by $\frac{n\eta}{1-n\eta} \sum_{j=1}^n |x_j|$.

ii. Define $T(i,k) = \sum_{j=i}^k x_j$ *to compute the summation over a range of indices. A recursive idea is to compute* $T(i,k)$ *by*

$$T(i,k) = T(i, \lfloor \frac{i+k}{2} \rfloor) \boxed{+} T(\lfloor \frac{i+k}{2} \rfloor + 1, k)$$

until the range only contains one index. If we apply this idea to the summation as $S_n = T(1,n)$, *show this summation method results in a rounding error bounded by* $\frac{H\eta}{1-H\eta} \sum_{j=1}^n |x_j|$ *with* $H = \lceil \log_2 n \rceil + 1$.

Problem 1.6. *As a followup of Problem 1.2. Now we assume that the evaluations of* ***cosine*** *and* ***sine*** *have rounding errors* [2]

$$fl(\cos(A)) = \cos(A)(1+\delta), \quad |\delta| \leq \eta.$$

$$fl(\sin(A)) = \sin(A)(1+\delta), \quad |\delta| \leq \eta.$$

i. Show that the truncation error

$$\left| \frac{f(\alpha+h) - f(\alpha)}{h} - f'(\alpha) \right| \leq \frac{h}{2}, \quad f(x) = \sin(x).$$

ii. Suppose the division $\frac{h}{2}$ *and the addition* $\alpha + \frac{h}{2}$ *have no rounding errors* [3]. *Show that the* ***total error*** *(truncation error and rounding error) between* $f_2(\alpha, h)$ *and* $f'(\alpha)$ *is bounded by*

$$\frac{h}{2} + ((1+\eta)^5 - 1)$$

[2] See the source file of sin and cos for a reference, the rounding error is actually around $0.55\,\texttt{ULP} = 1.1\eta$.

[3] It is often true if $h = 2^{-p}$ for a moderate size p.

iii. Suppose the addition $\alpha + h$ has no rounding error, that is, $\texttt{fl}(\alpha + h) = \alpha + h$. Show that the ***total error*** *between $f_1(\alpha, h)$ and $f'(\alpha)$ is bounded by*

$$\frac{h}{2} + \frac{2\eta(1+\eta)^2}{h} + (2+\eta)\eta$$

and find the minimum of this bound along with its corresponding h.

Problem 1.7. *(Optional) The evaluation of monomial $P_n(x) = x^n$ can be computed in several ways. The most naive way computes each term x^n by sequential multiplications*

$$P_n(x) = \texttt{fl}(x) \boxed{*} P_{n-1}(x).$$

There is another common pairwise algorithm.

$$P_n(x) = \begin{cases} P_{n/2}(x) \boxed{*} P_{n/2}(x) & n \text{ is even} \\ \texttt{fl}(x) \boxed{*} P_{(n-1)/2}(x) \boxed{*} P_{(n-1)/2}(x) & n \text{ is odd} \end{cases}$$

Roughly estimate the rounding errors of computing $P_n(x)$ using the above naive method and the pairwise algorithm. Here you may assume $x > 1$.

2 Programming Problems

Implement the following program tasks using your favorite programming language. The `Python` or `MATLAB` starter kit is available at GitHub Link. Follow the guidelines there for your submission.

Problem 2.1. *If you are using MATLAB, run the following commands write down what you see, and briefly explain why.*

(a). `eps`

(b). `realmax`

(c). `realmin`

(d). `1 + eps - 1`

(d). `1 + eps/2 - 1`

(e). `realmin/1e10`

(f). `realmin/1e16`

(g). `realmax*10`

If you are using Python, run the following commands write down what you see, and briefly explain why.

(a). `import sys;print(sys.float_info.epsilon)`

(b). `import sys;print(sys.float_info.max)`

(c). `import sys;print(sys.float_info.min)`

(d). `import sys;print(1 + sys.float_info.epsilon - 1)`

(d). `import sys;print(1 + sys.float_info.epsilon /2 - 1)`

(e). `import sys;print(sys.float_info.min/1e10)`

(f). `import sys;print(sys.float_info.min/1e16)`

(g). `import sys;print(sys.float_info.max*10)`

For other programming language users, please adjust the above commands to your language.

Problem 2.2. *The famous Archimedes' formula for π calculates the perimeters of regular polygons inscribing or circumscribing a circle of unit diameter. Starting from hexagon, $p_0 = \frac{1}{\sqrt{3}}$, the iterative formula can be written in two equivalent forms:*

$$p_{n+1} = \frac{\sqrt{1+p_n^2}-1}{p_n}$$

and

$$p_{n+1} = \frac{p_n}{1+\sqrt{1+p_n^2}}.$$

*where p_n is the length of each side of the regular polygon with $6*2^n$ sides. The approximation to π is computed by $s_n = 2^n * 6 * p_n$. Tabulate the error $|s_n - \pi|$ for various n using the above two iterative formulae. Explain the results in the **comments** of the submission.*

Problem 2.3. *Suppose $a, b \in \mathbb{F}$, the rounding error for the summation $s = \texttt{fl}(a+b)$ can be computed using the Kahan compensated summation*

$$\texttt{fl}(\texttt{fl}(s-a)-b),$$

where `fl` means rounding. Based on this property, one can keep track of the rounding error. Implement the Kahan compensated summation for computing $S = \sum_{j=1}^{n} x_j$, the algorithm is given below in Algorithm 1.

Algorithm 1: Kahan compensated summation

Data: $\{x_j\}_{j=1}^n \subset \mathbb{F}$
Result: $s_n = \sum_{j=1}^n x_j$
$j \leftarrow 1$, $e_j \leftarrow 0$, $s_j \leftarrow x_j$ // initialization;
while $j < n$ **do**
  $j \leftarrow j+1$
  $y_j = x_j - e_{j-1}$ //remove compensated error;
  $s_j = s_{j-1} + y_j$ //perform summation;
  $e_j = (s_j - s_{j-1}) - y_j$ //restore the rounding error;
end

Problem 2.4. *Compare your implemented Kahan summation and the built-in summation function `sum`() under single precision [4]. Summarize your findings (e.g. less/more accurate or comparable) in the **comments** of the submission.*

[4] https://www.mathworks.com/help/matlab/ref/single.html

2.1 Extra Problems for MATH 6630

Problem 2.5. *Implement the 2nd method in Problem 1.5 and compare its performance with the Kahan summation under single precision. Summarize your findings (e.g. less/more accurate or comparable) in the* ***comments*** *of the submission.*