Midterm 553.481/681, February 26, 2024

Do all **three** *of the following problems.* Show all your work. Answers without supporting work may receive no credit.

Students may discuss the exam **only** with the instructor and the teaching assistant. No discussion of the exam contents, directly or indirectly, is permitted among students or with any third parties. Any book or internet resource may be used, as long as the book or the website are cited, along with the material taken from it.

You may use any numerical software available, unless you are specifically instructed in the problem statement to write your own code. All codes written by you should be turned in with the exam, as a Matlab script or other executable computer program. Numerical results without the code that produced them will receive no credit.

I attest that I have completed this exam without unauthorized assistance from any person, materials, or device:

Full name: ____

Signature: ____

(See the Johns Hopkins Handbook *Academic Ethics for Undergraduates*).

Problem 1. IEEE floating-point arithmetic has levels of precision beyond single and double. All of these in standard layout have the first bit for the sign, the next r bits for the stored exponent SE, and the final p bits for the fraction F. The stored exponent is the true exponent E plus the bias, $SE = E + B$, where $B = 2^{r-1} - 1$. In particular, the following precision levels exist:

$$\begin{array}{lll} \text{quadruple:} & r = 15, & p = 112 \\ \text{octuple:} & r = 19, & p = 236 \end{array}$$

In hexadecimal format, quadruple-precision numbers are thus represented by strings of 32 hexadecimal digits, whereas octuple-precision numbers are represented by strings of 64 hexadecimal digits.

Consider the following two IEEE octuple-precision numbers in hexadecimal format:

(i) `40000921fb54442d18469898cc51701b839a252049c1114cf98e804177d4c762`
(ii) `bffff6a09e667f3bcc908b2fb1366ea957d3e3adec17512775099da2f590b066`

(a) Round both of these numbers to quadruple precision (in binary arithmetic or equivalently in hexadecimal) and give for both the hexadecimal representation of the IEEE quadruple-precision number in standard layout.

(b) Round both of the quadruple-precision numbers in (a) to double precision (once again in binary or hexadecimal arithmetic) and give for both the hexadecimal representation of the IEEE double-precision number in standard layout.

(c) Find the decimal representation of the double-precision numbers in (b) to 16 significant figures. You may use `hex2num` in Matlab to check your answer, but explain independently how you arrive at your answers.

Problem 2. Continuous functions on the closed interval $[a, b]$ can be assigned a norm

$$\|f\| = \max_{x\in[a,b]} |f(x)|$$

and we then say that a sequence $\{f_n\}$ of such functions converges to a function f, or $f = \lim_{n\to\infty} f_n$, if and only if

$$\lim_{n\to\infty} \|f_n - f\| = 0.$$

(a) If $I(f) = \int_a^b f(x)\,dx$ is the Riemann integral, then show that

$$|I(f)| \leq (b-a) \cdot \|f\|.$$

(b) Is integration well-posed for continuous functions on the closed interval $[a, b]$? In other words, is there a well-posed output $I(f)$ for input f?

(c) Consider the specific sequence of continuous functions $f_n(x) := \frac{1}{n}\sin(n^2 x)$ on the interval $[0, 2\pi]$. Show that $\lim_{n\to\infty} f_n = 0$. Is it true as well that $\lim_{n\to\infty} f_n' = 0$? Explain your answer.

(d) If we consider functions on the closed interval $[a, b]$ with a continuous derivative, then is differentiation well-posed for the stated norm? In other words, is there a well-posed output $D(f) = f'$ for input f?

Problem 3. Consider the following function

$$g(x) := x - \frac{2f(x)f'(x)}{\Delta(x)}, \quad \Delta(x) := 2(f'(x))^2 - f(x)f''(x) \tag{$*$}$$

such that the iteration $x_{n+1} = g(x_n)$ locally converges to x_* satisfying $f(x_*) = 0$.

(a) Show that this iteration has at least cubic order of convergence when $f'(x_*) \neq 0$ and when $f \in C^4$ near x_*.

(b) Write a code to implement iteration with the function (*), taking care to minimize the number of floating point operations in each iteration.

(c) Use your code to solve numerically for a root of the function $f(x) = e^x - 3x$ starting with $x_0 = 2$ and $TOL = 10^{-15}$. Compare with the Newton method for the same x_0 and TOL, both in terms of the number of iterations and the wall clock time required. In particular, calculate the ratio of the wall clock times for the two methods and explain this ratio quantitatively.

(d) Repeat part (d) for the function $f(x) = E_1(x) - x$. Note that

$$E_1(x) := \int_x^\infty \frac{e^{-t}}{t} dt$$

and this function can be evaluated with the Matlab function `expint`. In order to explain the ratio of clock times quantitatively, you will need to estimate the amount of time to evaluate the function $f(x)$ versus the time to evaluate $f'(x)$ or $f''(x)$.