University of Colorado
Department of Computer Science

Numerical Computation CSCI 3656

Spring 2015

Problem Set 1 Solutions

If there are any discrepancies between your answers and the solutions, please come to office hours to discuss them!

1. *[10 pts]* Using your favorite computer language, play around with some simple arithmetic calculations and find one whose answer is incorrect because of the imprecision of floating-point arithmetic. Submit the line of code, the correct answer (i.e., what you'd get if you were doing that calculation with pencil & paper, with infinite-precision numbers), the answer that the calculation produced on your computer, and an explanation of what it was that the computer arithmetic system did that made the answer wrong.

Answers will certainly vary. For example, typing $11^2 - (11^2 + 1 * 10^5)$ into MATLAB returned a value of $-1.000000000317414 * 10^{-5}$ instead of $-1 * 10^{-5}$. The issue here is what we discussed in class: computers don't store real numbers; rather, they store a number as the nearest "machine number." The error between the actual number and the number that is stored in the computer causes arithmetic problems like this.

2. *[6 pts]* Consider a variation of the IEEE standard in which the exponent is represented by seven bits and the fractional part of the mantissa is represented by four bits. What are the largest and smallest positive machine numbers that can be represented using this system, not including $\pm\infty$ or ± 0? Give your answers as both binary and decimal numbers. Assume that all other rules of the IEEE standard apply, including normalized numbers and reserved exponent values. Neglect subnormals (aka "graceful underflow"), which are permitted, but not *required*, by that standard.

The exponent bits here can run from 0000001 to 1111110 (i.e, 1 to 127 in decimal) because 0000000 and 1111111 are reserved for the purposes of representing ± 0 and $\pm\infty$. The bias in a system with seven exponent bits is $2^{7-1} - 1 = 63$, so the exponent can range from -62 to 63. The smallest positive number in this system, represented by the bit pattern 0 0000001 0000, is equivalent to $1.0000_2 \times 2^{-62}$, which is 2.1684×10^{-19} in decimal. (Don't forget the "leading one" normalization: there is understood to be a "1." in front of the fractional part of the mantissa that's given in the mantissa bits.) The largest positive number is 0 1111110 1111, which is equivalent to $1.1111_2 \times 2^{63}$, which is 1.7870×10^{19} in decimal.

4. *[6 pts]* Consider a variation of the IEEE standard in which the exponent is represented by e bits and the fractional part of the mantissa is represented by three bits. If the largest and smallest positive machine numbers in this system are 15.0 and 0.25, respectively, what is e? As in the previous problem, assume that all other rules of the IEEE standard apply.

In a system with e exponent bits, the allowed range is 1 to $2^e - 2$—not 0 to $2^e - 1$, as one might expect, because the "all zeroes" and "all ones" bit patterns are reserved. Then you have to factor in the bias in the system, which is $2^{e-1} - 1$ (127, in the case of the single-precision IEEE floating-point example used in class.) So the actual allowed range of the exponent is $1 - (2^{e-1} - 1)$ to $(2^e - 2) - (2^{e-1} - 1)$. In this *particular* system, the mantissa can run from

1.000_2 (which is the same thing as 1.000_{10}) to $1.111_2 = 1.875_{10}$. We don't know e, but know that the smallest number ($1 \times 2^{-2^{e-1}}$) is 0.25, so $e = 3$. In such a system, the exponent can range from -2 to 3, so the largest number will be 1.875 $times 2^3 = 15.0$. Sorry about the arithmetic bug in the first version!

5. *[6 pts]* Consider a computer that uses a really dumb arithmetic system that simply stores numbers in n-bit memory locations by dicing up the desired number range into $2^n - 1$ even-size chunks. If the range of numbers to be stored is -100000 to 100000 and $n = 5$, what is the (decimal) range of numbers that will be stored as 10000_2?

The range of numbers is $(0, 6451.6129)$. With $n = 5$, there are $(2^n) = 32$ possible bit patterns to be used for the number range -100000 to 100000. The lowest bit would represent a magnitude m of $(200000/31) \approx 6451.6129_{10}$. In this system, the bit pattern 00000_2 would represent -100000; in general, the bit pattern will represent $-100000 + (x * m)$, where x is the unsigned integer representation of the 5-bit pattern. 10000_2 represents the number $-100000 + (16 * 6451.6129) \approx 3225.8065$. If we assume that a number within -100000 to 100000 will be rounded to the closest bit representation, then the range of decimal numbers stored as 10000_2 is $(3225.8065 - m/2)$ to $(3225.8065 + m/2)$, which is $(0, 6451.6129)$ or $(0, m)$

6. *[4 pts]* The thermometer at the top of the Panoramic lift at Winter Park registered 10 degrees Fahrenheit last Saturday morning, but the true temperature was 9 degrees. What were the absolute and relative errors in that measurement?

The absolute error is one degree ($|9 - 10|$) and the relative error is $|9 - 10|/9$ or 11.1%.

7. A bit about roots

- *[2 pts]* Give an example of a function, $f(x)$, that has no roots. $f(x) = 6$
- *[2 pts]* Give an example of a function, $f(x)$, that has exactly one root. $f(x) = ax + b$ for pretty much any a and b.
- *[2 pts]* Give an example of a function, $f(x)$, that has an infinite number of roots. $f(x) = cos(x)$ or $f(x) = sin(x)$, for example.
- *[2 pts]* How many roots does $f(x) = x^2 - 4x + 6$ have? Are they real or complex? Two, and they are complex.