Assignment 1 - Floating Point Variables

Martik Aghajanian

1 Question 1

(a) For tabulating the function $f(x) = x^{10}$ within the domain $0 < x < 10^4$, one should use a double precision floating point variable, and not a single precision floating point. This is because the highest achievable single precision floating point variable is $1.11111111111...1 \times 2^{128} \sim 10^{38}$, where the mantissa is composed of twenty three 1's and the exponent is its maximum possible value. This does not reach the highest possible function output within this domain (10^{40}), so a single precision cannot be used. A double is able to achieve exponents which take the representation of the real numbers to much higher powers of 10 than this upper range bound of 10^{40}.

(b) For the function $f(x) = e^{x^2}$, the upper domain value produces a result of $f(10) = e^{100} \sim 10^{43}$, which will also require a double precision floating point variable to tabulate and evaluate. On an aside from this, the function could be managed in a long calculation using single precision floating point variables if instead we worked with the natural logarithm of the function, since there is a very largely increasing output with an increasing input within the domain $0 < x < 10$, however, this would nevertheless reduce the accuracy of our representation of this real number, were we to exponentiate it later, so to tabulate it we would still need a double precision floating point variable.

(c) The machine accuracy is the smallest floating point number we can add to 1.0 such that the result is still greater than 1.0. This is different to the smallest representable number and the smallest normalised floating point number because the machine accuracy indicates the size of the window of precision which is book-ended with $2^0 = 1$, and gives the largest size difference between floating point numbers which can be registered without the smaller of the numbers being negligible. This machine accuracy can be determined by half of the smallest possible value of the mantissa (with zero exponent) since rounding up from this number will a produce a change in the least significant digit of the mantissa by 1.

The smallest representable number gives the smallest element of the real number set for which a non-zero number can be given as a representation of it. This is essentially the smallest possible mantissa (unnormalised), times the base (2) to the power of the smallest possible exponent. Suppose the mantissa consisted of N bits, and the exponent had a minimum value of $-E_{min}$, then the smallest representable number would be $\pm 2^{-N+1} \times 2^{-E_{min}}$ ($-(N-1)$ since after 2^0 there are $N-1$ negative powers of two) which is unnormalised and represents the smallest step we can take on the real number line between consecutive elements of the real number set. This is different to the machine accuracy since the smallest representable number comes as a limitation on the size of the floating points we work with, whilst the machine accuracy comes as a limitation on the number of digits we can use to represent any number within the set of real numbers.

The smallest normalised number is generally larger than the smallest representable number because for a normalised number, there is always a 1 in the first bit of the mantissa (equivalent to the first digit of a number in base-ten standard form being both non-zero and the only digit left of the decimal point). This is so the floating point number always takes the form $1.n_1n_2n_3 \cdots n_N \times 2^E$. In comparison, the first bit of the smallest representable number is not 1. The smallest normalised number can be found by setting the mantissa to $1.000 \cdots 000$ and the exponent to its minimum value,

whilst the smallest representable number can be found by setting both the mantissa and the exponent to its minimum value. The difference between these two is that the smallest normalised number is the smallest real number we can give non-zero representation accurate to within the machine accuracy, whereas the smallest representable number is less accurate.

2 Question 2

The machine accuracy outputted from the program is, 1.19209×10^{-7} for single precision, 2.22045×10^{-16} for double precision, and 1.0842×10^{-19} for extended precision. The "extended" precision float in this test within C++ was the 80-bit "long double" precision available in C++ which consists of a 63-bit mantissa[1]. The values expected would be half the smallest possible value of the mantissa, which would be $2^{-(N-1)}/2 = 2^{-N}$ where N is the number of bits for the mantissa in that particular precision of floating point. This is so that rounding this up would produce a change of 1 in the mantissas least significant digit. For single precision the mantissa is 23 bits and hence the expected value is $2^{-23} = 1.19209 \times 10^{-7}$. For double precision the mantissa is 52 bits and the expected value of machine accuracy is $2^{-52} = 2.22045 \times 10^{-16}$. For extended precision (80-bit long double), the mantissa is 63 bits and thus the expected machine accuracy is $2^{-63} = 1.0842 \times 10^{-19}$. These are all consistent with the output of the program. I was not able to find a means of implementing a quadruple precision floating point in C++ with 128-bits without using an external library but I expect that theoretically the machine accuracy would be outputted as $2^{-112} = 1.9259 \times 10^{-34}$.

Ref [1] (mantissa of extended precision IEEE-754): http://blogs.perl.org/users/rurban/2012/09/reading-binary-floating-point-numbers-numbers-part2.html