CS / MATH 4334 : Numerical Analysis
Homework Assignment 1

Matthew McMillian
mgm160130@utdallas.edu

September 5, 2018

Theoretical Problems

1. Show how to evaluate the polynomial $p(x) = 2x^{25} + 7x^{15} - x^{10} + 4x^5 - 1$ using as few arithmetic operations as possible.

   To begin to solve this problem, we must first break down the polynomial p(x). I will start by storing some of the possible polynomial values ahead of time to conserve arithmetic operations.

$x = x$	- identity	(0+0 = 0 multiplications)
$x^2 = x * x$	- 1 multiplication	(0+1 = 1 multiplications)
$x^4 = x^2 * x^2$	- 1 multiplication	(1+1 = 2 multiplications)
$x^5 = x^4 * x$	- 1 multiplication	(2+1 = 3 multiplications)

   In total, storing values until we have stored an x^5 will net us a total of 3 multiplications. Next, we will apply a Horner's expansion to the function $p(x)$, given by:

   $$\begin{aligned}
   p(x) &= 2x^{25} + 7x^{15} - x^{10} + 4x^5 - 1 \\
   p(x) &= x^5 * (2x^{20} + 7x^{10} - x^5 + 4) - 1 \\
   p(x) &= x^5 * (x^5 * (2x^{15} + 7x^5 - 1) + 4) - 1 \\
   p(x) &= x^5 * (x^5 * (x^5 * (2x^{10} + 7) - 1) + 4) - 1 \\
   p(x) &= x^5 * (x^5 * (x^5 * (x^5 * (2x^5 + 0) + 7) - 1) + 4) - 1 \\
   p(x) &= x^5 * (x^5 * (x^5 * (x^5 * (x^5 * (2) + 0) + 7) - 1) + 4) - 1
   \end{aligned}$$

   Expanding $p(x)$ with Horner's method and storing the calculations prior to the functions evaluation nets us a total of $\boxed{\text{8 multiplications and 5 additions or subtractions}}$.

2. Convert the binary number $101101.000\overline{1011}$ to decimal form.

To convert this binary number to a decimal number, we must convert the items on both the LEFT and RIGHT of the radix point. From the LEFT, we use the algorithm to multiply the binary digits by their respective 2^x compliments:

$$101101_2 = (1 * 2^5) + (0 * 2^4) + (1 * 2^3) + (1 * 2^2) + (0 * 2^1) + (1 * 2^0) = 45_{10}$$

From the RIGHT, we could normally multiply by a base 2 number algorithm. However, the repeating decimal makes it tricky. In this case, we must normalize our decimal number and apply some algebra tricks to obtain and approximate value for the decimal.

$$x = .000\overline{1011}$$
$$2^3 x = .\overline{1011}$$
$$2^7 x = 1011.\overline{1011}$$
$$2^7 x - 2^3 x = (1011)_2 = (11)_{10}$$
$$x * (2^7 - 2^3) = 111$$
$$x * (120) = 11$$
$$x = \frac{11}{120} \approx .091\overline{6}$$

Thus our final base 10 decimal number is a concatenation of our LEFT and RIGHT numbers, which is $\boxed{45.091\overline{6}}$.

3. Consider the decimal number -26.1. Convert this number to binary form, then determine the machine representation of this number in double precision. Give the entire set in hexadecimal form.

To convert this decimal number to binary, we must convert both sides of the radix point using the divide and multiply algorithm:

Left of Radix Point	Right of Radix Point
$26 \div 2 = 13r0$	$.1 * 2 = .2r0$
$13 \div 2 = 6r1$	$.2 * 2 = .4r0$
$6 \div 2 = 3r0$	$.4 * 2 = .8r0$
$3 \div 2 = 1r1$	$.8 * 2 = 1.6r1$
$1 \div 2 = 0r1$	$.6 * 2 = 1.2r1$
	$.2 * 2 = 4r0$
	continuing...

Thus given by applying the algorithm, we obtain a binary number of $-11010.0\overline{0011}$ for part (a). Now, to determine the machine representation of this binary number in double precision, we must find out 3 things; the sign bit, the exponent bits, and the mantissa.

- The sign bit is easy. Since we have a negative number our sign bit is set to 1.
- For the exponent, we must first normalize the binary number, and add an offset of 1023 to our exponent to obtain the correct number for the computer:

  Given, e $= 11010.0\overline{0011} * 2^0$

  Then we normalize, e $= 1.101000\overline{0011} * 2^4$

  Then we add an offset to our exponent, $1023 + 4 = 1027$

  Then we convert to binary, $(1027)_{10} = (10000000011)_2$
- For the mantissa, we put in the numbers after our normalized radix point and check for rounding:

  Let m be the mantissa

  Given $m = |_0$ 1010 | 0001 | 1001 | 1001 | 1001 | ... | 1001 $_{52}|_{53}$ 1001 | ...

  Then we round m. Since the 53nd bit is a 1, and there are non-zero bits after it, we add a 1 to the 52nd bit.

  Thus our $m = |_0$ 1010 | 0001 | 1001 | 1001 | 1001 | ... | 1010 $_{52}|$
- To finalize the problem, we must concatenate our sign bit, exponent bits, and mantissa bits and convert the entire double precision expression to hexadecimal:

  Let $s, e, m =$ the sign bit, exponent bits, and mantissa respectively.

  Let $d =$ the concatenation of $s, e,$ and m, equivalently $d = s + e + m$.

  $d =$ 1100|0000|0011|1010|0001|1001|1001|1001|1001|1001|1001|1001|1001|1001|1010

  Converting to hexadecimal, $d =$ C03A1999999999A

Thus, our final binary number and double precision hexadecimal number are $\boxed{-11010.0\overline{0011}}$ and $\boxed{\text{C03A1999999999A}}$ respectively.

4. Suppose a certain computer stores decimal numbers (instead of binary) by chopping (instead of rounding) each normalized decimal number to 6 significant digits, i.e., $d_0.d_1d_2d_3d_4d_5$, where $d_0 \neq 0$. Find an upper bound on the relative error from this chopping.

   To find the relative error for this computer, we define relative error to be $|\frac{x-x_c}{x}|$, where x, x_c are an exact number and an approximate number respectively.

   Let $x = d_0.d_1d_2d_3d_4d_5 * 10^p$ be exact.

   Let $x_c = fl(x) = d_0.d_1d_2d_3d_4d_5 * 10^p$ be approximate.

   Then, our relative error becomes $|\frac{x-x_c}{x}| = |\frac{0.d_1d_2d_3d_4d_5}{9.d_1d_2d_3d_4d_5}|$.

   Then, we try to find an upper bound to our error by finding the maximum value of the numerator n_{max}, and minimum value of the denominator d_{min}, which are $n_{max} = 0.00009$ and $d_{min} = 9.00000$.

   This results in the relative error of $|\frac{0.00009}{9}| = 0.00001 = 10^{-6}$.

Therefore, similarly to the example in class, we determine that the upper bound of relative error for chopping roundoff error is 10^{-6}, or more generally, 10^{-p}, where p is the significant digits of your system.

5. Show how roundoff error can propagate through multiplication, given x_c, y_c are approximate values and x, y are exact values. Also, let $x_c = x + \epsilon$ and $y_c = y + \epsilon$.

   To find the roundoff error propagation, we must substitute in the values for x_c and y_c and solve to find a representation that has an error of $\frac{1}{2}\epsilon_{mach}$.