HW 1 Solutions

Math 472, Spring 2011

The homework was to work out exercises 3(a), 4(a), 6 and 7 (p. 15) from T. Sauer's "Numerical Analysis" text (a portion of Chapter 0 was passed out in class, and is temporarily available on CLEo).

3(a) Do the following sum by hand in IEEE double precision arithmetic using the rounding rule. Check your answers in Matlab. $(1 + (2^{-51} + 2^{-53})) - 1$

SOLUTION: First, to put the innermost number in floating point (normalized) form:

$$2^{-51} + 2^{-53} = 2^{-51}(1 + 0.01) = 1.01 \times 2^{-51}$$

Next, if we add one, we get the following. The space is after bit 52:

$$1.0000 \cdots 010 \;\; 1$$

To use the rounding rule, we see that this is the exceptional case (bit 53 is 1, and all zeros afterward)- Bit 52 is already zero, so truncate.

If we now subtract 1, we will have 2^{-51} left. Therefore, the end result is 2^{-51}. To check in Matlab:

```
>> (1+(2^(-51)+2^(-53)))-1

ans =
   4.4409e-16

>> 2^(-51)

ans =
   4.4409e-16
```

4(a) Same as last problem: $(1 + (2^{-51} + 2^{-52} + 2^{-54})) - 1$

SOLUTION: Very similar, except we won't have bit 53 equal to 1, so the rounding rule will change. First, the innermost value will be (in floating point normalized form):

$$2^{-51}(1 + 0.1 + 0.001) = 1.101 \times 2^{-51}$$

Next, add 1. Now (the space is after bit 52) we have:

$$1.000 \cdots 0011 \, 01000 \cdots$$

Now, the rounding rule says to truncate since bit 53 is zero. Therefore, the number is (finishing at bit 52):

$$1.0000 \cdots 00011$$

Subtract 1 and that will leave us with (base 10):

$$2^{-51} + 2^{-52} = 2^{-52}(2+1) = 3 \times 2^{-52}$$

Verify in Matlab:

```
>> (1+(2^(-51)+2^(-52)+2^(-54)))-1

ans =
   6.6613e-16

>> 3*(2^(-52))

ans =
   6.6613e-16
```

6. This question has a couple of parts:

   - Is $1/3 + 2/3$ exactly equal to 1 in double precision floating arithmetic, using the rounding rule?

     SOLUTION: First, in base 2, $1/3 = 0.0101\cdots$ and $2/3 = 0.101010\cdots$. Now, as normalized floating point numbers (and using the rounding rule), we re-write $1/3$ as (the space is after bit 52)

     $$1.010101\cdots01 \;\; 0101\cdots \times 2^{-2} \quad \Rightarrow \quad 1.0101\cdots01 \times 2^{-2}$$

     Bit 53 is zero, so we truncate. The same things happens with $2/3$:

     $$1.010101\cdots01 \;\; 0101\cdots \times 2^{-1} \quad \Rightarrow \quad 1.0101\cdots01 \times 2^{-1}$$

     Now to add, make both numbers have the same exponential part, then apply the rounding rule:

     $$\begin{array}{rr} 1.0101\cdots0101 \;\; 0 & \times 2^{-1} \\ 0.1010\cdots1010 \;\; 1 & \times 2^{-1} \\ \hline 1.1111\cdots1111 \;\; 1000 & \times 2^{-1} \end{array} \quad \Rightarrow \quad 10.000000\cdots0 \times 2^{-1} = 2 \cdot \frac{1}{2} = 1$$

     ANSWER: Yes, $1/3 + 2/3 = 1$ in floating point arithmetic.

   - Does this help explain the rule as it is? Somewhat.
   - Would the sum be the same if chopping were used? No. The sum would be

     $$1.1111\cdots1 \times 2^{-1}$$

     And we showed that this is 2^{-52} more than the previous answer (that is, adding this number to 2^{-52} gives the value 1).

7. The same technique that was applied earlier is used again in Exercise 7:

   (a) First we compute $(7/3 - 4/3) - 1$ and show that it gives you ϵ_{mach}: First, after rounding, the floating point forms of 7/3 and 4/3 are:

   $$fl(7/3) = 1.001010\cdots101011 \times 2^1 \qquad fl(4/3) = 1.010101\cdots0101 \times 2^0$$

   Therefore, subtracting them gives:

   $$\begin{array}{rll} & 1.001010\cdots101011\ \ 00 & \times 2^1 \\ - & 0.101010\cdots101010\ \ 10 & \times 2^1 \\ \hline = & 0.100000\cdots000000\ \ 1 & \times 2^1 \end{array} \quad \Rightarrow \quad 1.0000\cdots0001 \times 2^1$$

   which is $1 + \epsilon_{\text{mach}}$. After subtracting 1, we get ϵ_{mach}.

   (b) Next, we show that $(4/3 - 1/3) - 1$ gives you zero.
   Subtracting the floating point forms:

   $$\begin{array}{rll} & 1.010101\cdots010101\ \ 00 & \times 2^0 \\ - & 0.010101\cdots010101\ \ 01 & \times 2^0 \\ \hline = & 0.111111\cdots111111\ \ 11 & \times 2^0 \end{array} \quad \Rightarrow \quad 1.0000\cdots000 \times 2^0$$

   after applying the rounding rule. Therefore, we get zero after subtracting 1.