Math 128A, Summer 2019
Lecture 2, Tuesday 6/25/2019

1 Review of Lec 1 Floating Point Arithmetic

In last lecture, we ended with rounding FPN (Floating Point Numbers), and tie-breaking.

Definition: Normal number -

$$[S] \qquad [C] \qquad [f]$$

$$(-1)^s \cdot 2^{(C-1023)} \cdot (1+f)$$

(65-bit effective by adding the implied 1 in mantissa)

Definition: Subnormal number -

$$C := 0$$

$$\implies (-1)^S \cdot 2^{-1022} \cdot (f)$$

"Gradual underflow". We lose the $1 + \cdots$, but we get another 51 bits before our numbers go to zero $2^{-1023} \to 2^{-1022}$.

Note: We used to use "fixed-point" arithmetic (with all non-sign bits assigned to the mantissa; i.e. no characteristic value) back when computers were vacuum tubes in a room. Now with the characteristic, we get better usability.

Other special cases:

Consider that ± 0 are both "standard numbers". "$\pm$Inf" as well. But there's NaN (not a number), which is anything that has no valid representation.

1.1 Spacing of FPNs

Definition: Machine Epsilon (mach eps) -
Also called **ULP** for "unit in the last place".

$$\varepsilon := 2^{-52}$$

So we have the following spacing:

$$1, 1+\varepsilon, 1+2\varepsilon, \ldots, 2-\varepsilon, 2, 2+2\varepsilon, 2+4\varepsilon, \ldots, 4-2\varepsilon, 4, \cdots$$

Example: Constructing the IEEE FP representations:

$$\begin{aligned}
1 &= S = 0, C = 01111111111 = 1023, f = 0\cdots 00 \\
2 &= S = 0, C = 10000000000 = 1024, f = 0\cdots 00 \\
2+2\varepsilon &= S = 0, C = 10000000000 = 1024, f = 0\cdots 01 \\
2+4\varepsilon &= S = 0, C = 10000000000 = 1024, f = 0\cdots 10 \\
2+6\varepsilon &= S = 0, C = 10000000000 = 1024, f = 0\cdots 11 \\
1-\frac{\varepsilon}{2} &= S = 0, C = 1022, f = 1\cdots 1
\end{aligned}$$

Definition: Rounding -

Rounding: $fl : \mathbb{R} \to$ nearest FPN

Example:

$$\text{fl }(1+\varepsilon+\frac{\varepsilon}{\pi}) = 1+\varepsilon\text{fl }(1-\frac{\varepsilon}{\pi})$$

Resolving ties: (to last bit zero)

$$\begin{aligned}
&fl\left(1+\frac{\varepsilon}{2}\right) = 1 \\
&fl\left(1-\frac{\varepsilon}{2}\right) = 1-\frac{\varepsilon}{2} \\
&fl\left(1-\frac{\varepsilon}{4}\right) = 1 \quad \text{(this is more interesting)}
\end{aligned}$$

Note that a real number will live in between two FPNs, one will be even and one will be odd. It's not necessarily rounding down; just rounding to 0. This prevents bias accumulating ("drift") over many operations.

Remark: Rounding is "monotone", which means it preserves order (inequalities).

$$a < b \implies fl(a) \le fl(b)$$

Note the possible equality if they both round to the same FPN.

Aside/Motivation: debugging a program to run on supercomputer

$$\arccos(x); x = \frac{a}{a^2+b^2}$$

But now with the IEEE standard, we don't have these issues with inequalities, where values go out of 'bounds'.

2 Rounding Error

$$\begin{aligned}
1 \le x < 2: \qquad |fl(x)-x| &\le \varepsilon \\
&\le \frac{\varepsilon}{2} \\
&\le \frac{\varepsilon}{2}|x| \\
\text{Relative Error} =: \quad \frac{|fl(x)-x|}{|x|} &\le \frac{\varepsilon}{2} \le \varepsilon
\end{aligned}$$

But of course, if we have a large x, our precision is much smaller because we need to use bits to store the size.

If we have the following addition:

$$\begin{aligned}
&0.00\overbrace{1\cdots1}^{52} \\
&+1.000\cdots0 \\
&=1.\underbrace{001\cdots1|11}_{54} = 1.010\cdots0 \\
&\qquad = [2^{-3}\cdot(1-\varepsilon)] + [1] = \underbrace{1 + 2^{-3} - 2^{-3}\varepsilon}_{\text{exactly}} \\
fl(x,y) &= fl(1 + 2^{-3} - 2^{-3}\varepsilon) = 1 + 2^{-3}
\end{aligned}$$

Mantra of FP Arithmetic: Let x, y be FPN, floating point numbers. The floating point result of any binary operation on two FPNs gives us:

$$fl(x+y) = \text{the exact result, correctly rounded}$$

And likewise for $fl(xy), fl(x-y), fl(x/y), fl(\sqrt{x})$.

So instead of doing FP arithmetic per term, we just perform the regular math, then round at the end.

Example: $0.00\underbrace{11\cdots11}_{52} = 2^{-3}\times \underbrace{1.\overbrace{\underbrace{1\cdots1}_{52}}^{f}}_{2-\varepsilon}$

Or equivalently, $fl(1+2^{-3}(2-\varepsilon)) = fl(1+2^{-2}-2^{-3}\varepsilon) = 1.01$.

So we have:

Definition: Relative Error $|\delta| \le \frac{\varepsilon}{2}$ -

$$\begin{aligned}
\frac{|fl(x) - x|}{|x|} &\le \frac{\varepsilon}{2} \\
|\underbrace{fl(x) - x}_{\delta x}| &\le \frac{\varepsilon}{2}|x| \\
fl(x) - x &= \delta x \\
fl(x) &= x + \delta x \\
&= x(1+\delta), \quad |\delta| \le \frac{\varepsilon}{2}
\end{aligned}$$

Remark: δ is INDEPENDENT of x.

Thus:

$$\begin{aligned}
fl(x \pm y) &= (x \pm y)(1+\delta), \qquad |\delta| \leq \frac{\varepsilon}{2} \\
fl(\sqrt{x}) &= \sqrt{x}(1+\delta), \qquad |\delta| \leq \frac{\varepsilon}{2} \\
fl((x+y)+z) &\neq (x+y+z)(1+\delta) \\
&= fl[fl(x+y)+z] \\
&= fl[(x+y)(1+\delta)+z] \\
&= [(x+y)(1+\delta_1)+z](1+\delta_2) \\
&= (x+y)(1+\delta_1)(1+\delta_2)+z(1+\delta_2) \\
&= (x+y)(1+\delta_1+\delta_2+\delta_1\delta_2)+z(1+\delta_2) \\
&= (x+y)(1+2\delta_3)+z(1+\delta_2), \qquad |\delta_3| \leq \frac{3}{2} \\
&= x(1+2\delta_3)+y(1+2\delta_3)+z(1+\delta_2)
\end{aligned}$$

Note that these deltas can be different. But each of these terms are very close to their respective true values x, y, z.

Definition: Backwards Error Analysis -
Take $fl(x+y+z) = \hat{x}+\hat{y}+\hat{z}$ where:

$$\begin{aligned}
\hat{x} &:= x(1+2\delta_3) \\
\hat{y} &:= y(1+2\delta_3) \\
\hat{z} &:= z(1+\delta_2)
\end{aligned}$$

As opposed to...

Definition: Forward Error Analysis -

$$\begin{aligned}
\frac{|fl(x+y+z)-(x+y+z)|}{|x+y+z|} &\leq \frac{|(x+y)(1+2\delta_3)+z(1+\delta_2)-(x+y+z)|}{|x+y+z|} \\
&\leq \frac{2|\delta_3||x+y|+|z||\delta_2|}{|x+y+z|} \\
&\leq \frac{2|x+y|+|z|}{|x+y+z|}\frac{\varepsilon}{2}
\end{aligned}$$

Example: Take $x = 1+\varepsilon, y = -\varepsilon, z = -1$. Recall that we had $1 \leq x < 2$, with $|fl(x)-x| \leq \frac{\varepsilon}{2}|x|$.

Our relative error bound is

$$\frac{2|x+y|+|z|}{|x+y+z|}\frac{\varepsilon}{2} \leq \frac{3}{0} \cdot \frac{\varepsilon}{2}$$

But we have division by zero, which is useless, so our result would be unverifiable.

Mantra 2 of Computing: Don't compute 0 by subtracting nonzero objects (because then we can't tell the relative error).

Key Takeaway:

$$fl(x) = x(1+\delta), \qquad |\delta| \leq \frac{\varepsilon}{2}$$

Example of large number of operations, backwards error analysis:

Suppose we want to find:

$$S_n = \sum_{j=1}^{n} x_j$$

We need an algorithm:

$$S_1 = x_1 \tag{1}$$
$$S_2 = S_1 + x_2 \tag{2}$$
$$\vdots \tag{3}$$
$$S_n = S_{n-1} + x_n \tag{4}$$

Pseudocode

```
S_n = S_{n-1} + x_n
unless
n = 1    when S_1 = x_1
recursion
```

In numerical analysis we tend to avoid recursion because it can be expensive (number of operations).

We conclude:

$$\begin{aligned} fl(S_n) = x_1[1+(n-1)\delta_1] \\ &+ x_2[1+(n-1)\delta_2] \\ &+ x_3[1+(n-2)\delta_3] \\ &+ \cdots + x_n[1+\delta_n] \end{aligned}$$

or more sloppily (overestimating),

$$\begin{aligned} fl(S_n) = x_1[1+(n)\delta_1] \\ &+ x_2[1+(n-1)\delta_2] \\ &+ x_3[1+(n-2)\delta_3] \\ &+ \cdots + x_n[1+\delta_n] \end{aligned}$$

So BEA (backwards error analysis) says that x has n errors.

If we want FEA (forward error analysis) and its diagnostic, then we look at:

$$\left| \frac{fl(S_n) - S_n}{S_n} \right| \leq \frac{n|x_1| + (n-1)|x_2| + \cdots + 1|x_n|}{|S_n|} \cdot \frac{\varepsilon}{2}$$

Remark: Notes end here for today.

Big ideas: BEA, FEA, $(1+\delta_n)$ and simplifications for compounding error from a sequence of operations.