Physics 305: Roundoff error

Vasileios Paschalidis[1]
[1] *Departments of Astronomy & Physics, University of Arizona, Tucson*

This note provides a basic understanding of what is roundoff error.

I. REPRESENTATION OF REAL NUMBERS ON THE COMPUTER

Real numbers are not fully represented on computers. The IEEE standard for represented 64-bit floating point numbers (a double in C/C++) is

$$(-1)^s 2^{(c-1023)}(1+f), \tag{1}$$

where s is called the sign, c the characteristic and f the mantissa. The representation of a real number is

$$|\overbrace{\underline{1\text{bit}}}^{\text{sign}}|\overbrace{\underline{\quad 11\text{bits}\quad}}^{\text{characteristic}}|\overbrace{\underline{\quad 52\text{bits}\quad}}^{\text{mantissa}}| \tag{2}$$

Of course s can be 0 or 1. When it is 0 we represent positive real numbers, when it is 1 we represent negative real numbers. For example, we can have the number (in binary)

$$|\underline{0}|\underline{10000000011}|\underline{101110010001}\overbrace{0000000000000000\ldots 0}^{40\times}| \tag{3}$$

which has

$$\begin{aligned} s =& 0 \\ c =& 1\times 2^{10}+0\times 2^9+0\times 2^8+\ldots 1\times 2^1+1\times 2^0 = 1027, \\ f =& 1\times\left(\frac{1}{2}\right)^1+0\times\left(\frac{1}{2}\right)^2+1\times\left(\frac{1}{2}\right)^3+1\times\left(\frac{1}{2}\right)^4+1\times\left(\frac{1}{2}\right)^5+0\times\left(\frac{1}{2}\right)^6+0\times\left(\frac{1}{2}\right)^7 \\ &+1\times\left(\frac{1}{2}\right)^8+0+0+0+1\times\left(\frac{1}{2}\right)^{12}+0+\ldots+0\times\left(\frac{1}{2}\right)^{52} = 0.722900390625 \end{aligned} \tag{4}$$

and hence the real number represented is

$$N_0 = (-1)^0 2^{(1027-1023)}(1+0.722900390625) = 27.56640625 \tag{5}$$

Now, we can start to see the limitations of floating point arithmetic.
The next largest real number that can be represented is

$$|\underline{0}|\underline{10000000011}|\underline{101110010001}\overbrace{0000000000000000\ldots 0\,1}^{39\times}| \tag{6}$$

This number has the same s, the same c, f which is larger by $1/2^{52} = 2.220446e{-}16$, i.e., $f = 0.7229003906250002204$, and hence the number represented is

$$N_1 = (-1)^0 2^{(1027-1023)}(1+0.7229003906250002204) = 27.566406250000355264 \tag{7}$$

The next smallest real number that can be represented is

$$|\underline{0}|\underline{10000000011}|\underline{101110010000}\overbrace{1111111111111111\ldots 1}^{40\times}| \tag{8}$$

This number, let's call it N_{-1}, has the same s, the same c, and f which is smaller by $-1/2^{12}+\sum_{n=13}^{n=52}1/2^n = 1/2^{52}$. Thus, N_0 represents half of the numbers between N_0 and N_1 and half of the numbers between N_{-1} and N_0. These examples explictly show that there is finite number of reals that is represented.

A. Overflow/Underflow

In addition, there is a finite range of numbers that can be represented. In particular, the largest possible positive real number corresponds (by convention) to $c = 2046$, $f = 1 - 1/2^{52}$, thus

$$N_{\max} = 2^{1023}(2 - 1/2^{52}) \approx 0.17977 \times 10^{309} \tag{9}$$

The smallest possible positive real number corresponds to $c = 1$, $f = 0$, thus

$$N_{\min} = 2^{-1022}(1 + 0) \approx 0.22251 \times 10^{-307} \tag{10}$$

If during a computation a number becomes greater than $N_{\max}$, we say we have overflow, and if a number becomes smaller than $N_{\min}$, we say we have underflow.

B. Decimal Machine Numbers

Let the machine numbers be represented in the normalized decimal floating-point form

$$\pm 0.d1d2\ldots d_k \times 10^n,\ 1 \le d1 \le 9, 0 \le d_i \le 9,\ i = 2, \ldots k \tag{11}$$

Numbers of this form are called k-digit decimal machine numbers.

Any positive real number

$$y = 0.d1d2\ldots d_k d_{k+1} d_{k+2} \ldots \times 10^n \tag{12}$$

will have a floating-point representation by terminating the mantissa at k decimal digits either by chopping or rounding.

Chopping means remove $d_k d_{k+1} d_{k+2} \ldots$ and produces the floating point k-digit number

$$fl(y) = 0.d1d2\ldots d_k \ldots \times 10^n \tag{13}$$

Rounding adds $5 \times 10^{n-(k+1)}$ and chops the result to obtain a number $fl(y) = 0.\delta_1\delta_2\ldots\delta_k$:

- If $d_{k+1} \ge 5$ then $d_k \to d_k + 1$ (round up).
- If $d_{k+1} < 5$ then chop off all but first k digits (round down).

When rounding down $\delta_i = d_i$. When rounding up the digits can change.

Exercize: consider $\pi = 3.14159265 = 0.314159265 \times 10^1$. What is the number that is represented after 5-digit rounding or chopping?

Answer: Using 5-digit chopping $fl(\pi) = 0.31415 \times 10^1 = 3.1415$. Using 5-digit rounding $fl(\pi) = \text{chop}[(0.314159265 + 0.00001) \times 10^1] = \text{chop}[(0.314160265) \times 10^1] = 0.31416 \times 10^1 = 3.1416$.

Definition: If $\tilde{p}$ represents an approximation to a real number p, the relative error

$$\frac{|p - \tilde{p}|}{p} \equiv \text{ Roundoff error.} \tag{14}$$

In the previous exersize, what is the roundoff error after chopping? What is the roundoff error after rounding?

Exercize: Think about the way real numbers are represented in a computer, i.e., Eqs. (1)-(5). Based on this, what is the roundoff error in 64-bit floating point arithmetic?

Answer: The smallest possible change to any real number comes from the mantissa term $1/2^{52} = 2.2204 \times 10^{16}$. Thus, this term represents approximately the roundoff error of 64-bit floating point arithmetic, i.e., 64-bit floating real numbers are accurate to about 16 digits.