Floating Point Numbers

Most numbers cannot be represented in a computer. Those that are not representable are approximated by a relatively small few that are. We will let the floating point approximation of x be called the *float* of x and write it as $\mathrm{fl}(x)$. A floating point number represents all of the reals in an interval near it. We can bound the length of this interval, and therefore the error that is made when approximating a number by its float. We assume that (normalized) floating point numbers have the form

$$\bar{x} = \pm(0.b_1 b_2 \ldots b_t)_2 \times 2^e, \quad \text{where} \quad e_n \le e \le e_p \ \text{ and } \ b_k \text{ is 0 or 1, but } b_1 = 1.$$

Think of it as a (base-2) fraction times 2^e. Numbers too |large| for this representation are said to *overflow*, and numbers too |small| are said to *underflow*. The set of real numbers which do not underflow or overflow is called the *floating point range* (FPR).

Since we have allotted t bits for the fractional part, the distance between $\bar{x}$ and its |larger| neighboring float is 2^{e-t}. Dividing this by $\bar{x}$ gives an upper bound on the relative distance between any two floats, the *machine epsilon*: $\epsilon_{mach} = 2^{1-t}$. We define the *unit round-off*, $\boldsymbol{\mu}$, to be half of this quantity: For a (binary) floating point system with a t bit fractional part, the unit round-off is $\boldsymbol{\mu} = 2^{-t}$ (with base β, $\boldsymbol{\mu} = \frac{1}{2}\beta^{1-t}$, so e.g. 4-decimal digit arithmetic ($t = 4$) has $\boldsymbol{\mu} = \frac{1}{2}10^{-3}$).

The Floating Point Representation Theorem.
Suppose x is a real number in the floating point range (x doesn't underflow or overflow). Then

$$\mathrm{fl}(x) = x(1+\epsilon), \quad \text{where} \quad |\epsilon| \le \boldsymbol{\mu}$$

This is a statement about relative error, and can equivalently be written as

$$\frac{|x - \mathrm{fl}(x)|}{|x|} \le \boldsymbol{\mu}.$$

Unfortunately, the set of floats is not closed under arithmetic operations. For example, when we add two floats, the result is not necessarily a float, but will instead be rounded to its float. Computers today follow an industry standard called the IEEE 754, which among many other things guarantees the following:

The Fundamental Axiom of Floating Point Arithmetic.
Let $x \ op \ y$ be some arithmetic operation. That is, *op* is one of $+$, $-$, $\times$, or $\div$. If x and y are (normalized) floats and $x \ op \ y$ is in the floating point range, then

$$\mathrm{fl}(x \ op \ y) = (x \ op \ y)(1+\epsilon), \quad \text{where} \quad |\epsilon| \le \boldsymbol{\mu}$$

The geometry is simple: When doing a single arithmetic operation with floats, we get the float which is closest to the true value (as long as it is in FPR). But be careful: this is a statement about floats; other numbers need to be rounded to floats before we can do any arithmetic!