## 2.7.2 IEEE Floating Point Arithmetic

To build a solid, practical understanding of finite precision computation, we set aside our toy, motivational base-10 calculator and consider the key ideas behind the widely accepted IEEE floating point standard. The IEEE standard includes a 32-bit single format and a 64-bit double format. We will illustrate concepts using the latter as an example because typical accuracy requirements make it the format of choice.

The importance of having a standard for floating point arithmetic that is upheld by hardware manufacturers cannot be overstated. After all, floating point arithmetic is the foundation upon which all of scientific computing rests. The IEEE standard promotes software reliability and enables numerical analysts to make rigorous statements about computed results. Our discussion is based on the excellent book by Overton (2001).

The 64-bit double format allocates a single bit for the sign of the floating point number, fifty-two bits for the mantissa , and eleven bits for the exponent:

$$x : \boxed{\pm} \boxed{a_1a_2 \ldots a_{11}} \boxed{b_1b_2 \ldots b_{52}} \tag{2.7.1}$$

The "formula" for the value of this representation depends upon the exponent bits:

If $a_1 \ldots a_{11}$ is neither all 0's nor all 1's, then $x$ is a *normalized* floating point number with value

$$x \;=\; \pm(1.b_1b_2 \ldots b_{52})_2 \;\times 2^{(a_1a_2 \ldots a_{11})_2 - 1023}. \tag{2.7.2}$$

The "1023 *bias*" in the exponent supports the graceful inclusion of various "unnormalized" floating numbers which we describe shortly. Several important quantities capture the finiteness of the representation. The *machine epsilon* is the gap between 1 and the next largest floating point number. Its value is $2^{-52} \approx 10^{-16}$ for the double format. Among the positive normalized floating point numbers, $N_{min} = 2^{-1022} \approx 10^{-308}$ is the smallest and $N_{max} = (2 - 2^{-52})2^{1023} \approx 10^{308}$ is the largest. A real number $x$ is within the *normalized range* if $N_{min} \le |x| \le N_{max}$.

If $a_1 \ldots a_{11}$ is all 0's, then the value of the representation (2.7.1) is

$$x \;=\; \pm(0.b_1b_2 \ldots b_{52})_2 \;\times 2^{(a_1a_2 \ldots a_{11})_2 - 1023} \tag{2.7.3}$$

This includes 0 and the *subnormal* floating point numbers. This feature creates a uniform spacing of the floating point numbers between $-N_{min}$ and $N_{min}$.

If $a_1 \ldots a_{11}$ is all 1's, then the encoding (2.7.1) represents `inf` for $+\infty$, `-inf` for $-\infty$, or `NaN` for "not-a-number." The determining factor is the value of the $b_i$. (If the $b_i$ are not all zero, then the value of $x$ is `NaN`.) Quotients like 1/0, -1/0, and 0/0 produce these special floating point numbers instead of prompting program termination.