2.8 Floating point numbers and round-off errors.

Round-off errors are due to the fact that people, calculators, and computers usually do not keep track of or store numbers exactly in the course of a series of calculations. Scientific and engineering computations are often done with numbers expressed in floating point form. A number x is expressed in *decimal floating point* form if it is written as a signed number with magnitude between 1 and 10 multiplied by an integral power of 10. In other words we write $x = \pm\, d_1.d_2\cdots d_j\cdots \times 10^q$ where the d_j are decimal digits with $d_1 \neq 0$.

Example 1.

168,500	has floating point representation	1.685×10^5
0.0378462	has floating point representation	3.78462×10^{-2}
- 0.00746	has floating point representation	$-\ 7.46 \times 10^{-3}$
1/3	has floating point representation	$3.333\ldots. \times 10^{-1}$

The number $\pm\, d_1.d_2\cdots d_j\cdots$ is called the *mantissa* while the power q is called the *exponent*. Actually, computers often use base 2 for their representation of floating point numbers, but most of the important issues with round off errors are present with base 10. So, for simplicity, we restrict our attention to base 10.

Note that q is the largest integer such that $10^q \leq |\, x\, |$ which implies

(1) $$q \;=\; \lfloor\, log_{10}\, |\, x\, |\, \rfloor$$

Then the mantissa $\pm\, d_1.d_2\cdots d_j\cdots$ is equal to $10^{-q}x$. Here $\lfloor\, y\, \rfloor$ denotes the floor of y which is the largest integer not exceeding y.

Note that in the case of $1/3 = 3.333\ldots. \times 10^{-1}$ one needs an infinite number of digits in the mantissa to represent 1/3 exactly. It is usually impossible to keep track of an infinite number of digits in the course of a series of computations, so people and computers usually do calculations keeping only a certain fixed number of digits of the mantissa at each step. This is called the *number of digits of precision* in the computations. Much of today's software does computations with at least 15 decimal digits of precision. We will assume a number x is rounded to x_a, although some computers chop x to get x_a.

Example 2. If a computation is done is using seven decimal digits of precision, then the number $x = 1/3$ would be approximated by $x_a = 3.333333 \times 10^{-1} = 0.3333333$. The absolute error between the number $x = 1/3$ and $x_a = 0.3333333$ is $1/3 \times 10^{-7}$ and the relative error is 10^{-7}.

In general, the term *round-off error* refers to the error that one makes by replacing a number with its floating point approximation rounded off to a certain number of digits. The size of a round-off error will vary. However, there is a close connection between the relative error of the round-off error and the number of digits of precision; see Proposition 1 below.

To make this more precise suppose $x = \pm d_1.d_2\cdots d_j\cdots \times 10^q$ in floating point form. Then if $x > 0$

$$(2) \qquad round(x, p) = \begin{cases} d_1.d_2\cdots d_{p-1}d_p \times 10^q & \text{if } d_{p+1} < 5 \\ d_1.d_2\cdots d_{p-1}(d_p+1) \times 10^q & \text{if } d_{p+1} \geq 5 \text{ and } d_p < 9 \\ d_1.d_2\cdots d_{s-1}(d_s+1)0\cdots 0 \times 10^q & \text{if } d_{p+1} \geq 5 \text{ and } d_s < 9 \text{ and } d_{s+1} = \cdots = d_p = 9 \\ 1.0\cdots 0 \times 10^{q+1} & \text{if } d_{p+1} \geq 5 \text{ and } d_1 = \cdots = d_p = 9 \end{cases}$$

denotes x rounded to p decimal places. If $x < 0$ then $round(x, p) = -\, round(-x, p)$.

Example 3. If $x = 1.685 \times 10^5$ and we round x to 3 decimal places we get 1.68×10^5.

On most calculators and computers the numbers are rounded-off to the same number of digits after each operation and to say that a computation is done with p decimal digits of precision means that the inputs and the result of each arithmetic operation are rounded to p digits. There is the following connection between the number of significant decimal digits and a bound on the relative error.

Proposition 1 Suppose $x = \pm d_1.d_2\cdots d_j\cdots \times 10^q$ in floating point form and let $x_a = round(x, p)$ be the approximation to x obtained by rounding x to p decimal places. Then the absolute error is no more than $5 \times 10^{q-p}$ and both relative errors ε_t and ε_a are no more than 5×10^{-p}.

Proof. We shall suppose x is positive; the case where x is negative follows from the case where x is positive and the fact that $round(x, p) = -\, round(-x, p)$. One has $\delta = | x - x_a | \leq 0.0\cdots05 \times 10^q$ where there are p-1 zeros between the decimal point and the 5. This is because we decrease x by no more than this amount to get x_a if we round x down to get x_a and we increase x by no more than this amount if we round up. Note that $0.0\cdots05 \times 10^q = 5 \times 10^{q-p}$. So $\delta \leq 5 \times 10^{q-p}$. Also $10^q \leq | x |$ and $10^q \leq | x_a |$. So $\varepsilon_t = \delta / | x | \leq 5 \times 10^{q-p} / 10^q = 5 \times 10^{-p}$, and similarly for ε_a. //

$round(x, p)$ can be expressed in terms of the chop operation.

$$chop(x, p) = d_1.d_2\cdots d_p \times 10^q = 10^{q-p+1} \lfloor 10^{p-q-1} x \rfloor$$

denotes x chopped off to p digits. If $x < 0$ then $chop(x, p) = -\, chop(-x, p)$. If $x > 0$ then

$$round(x, p) = chop(x + (5 \times 10^{q-p}), p)$$

In analysis of round-off errors it is often convenient to work with the machine $\boldsymbol{\varepsilon}$.

(3) *Machine* $\boldsymbol{\varepsilon}$ = the smallest number which when added to 1 using the given computational method gives a result larger than 1.

If the computations are done with p decimal digits of precision then $\boldsymbol{\varepsilon} = 5 \times 10^{-p}$. Note that Proposition 1 says that the relative error between a number and the approximation obtained by rounding it off to p decimal places is no more than $\boldsymbol{\varepsilon}$. The round-off error in storing a measured value in the computer is usually much smaller than the error in measurement. However, it is possible for the round-off error in arithmetic computations to be larger than the error due to the error in measurement. We shall see some examples in the next section.