Rounding Examples

Given $x \in \mathbf{R}$ we denote by x^* the nearest floating point approximation to x. In the case of a tie, x^* is chosen to be the approximation in which the least significant digit in the mantissa is even. In base 10, this is called the banker's round. When working with base 2 this rule ensures that the least significant bit of x^* is zero in case of a tie.

The IEEE 754 standard actually describes 4 types of rounding. For default, it specifies the rounding method we have denoted by x^*. This method has the advantage that rounding errors tend to cancel out. In particular, a sum of such rounding errors can be viewed as a random walk with equal chances for adding either positive or negative errors.

There are certain cases where rounding $x \in \mathbf{R}$ to the nearest floating point representation x^* is not desirable. For example, suppose we wanted an upper bound on x. Then computing an upper bound on x^* might not yield an upper bound on x because it could happen that $x^* < x$. To address these issues, IEEE 754 specifies three other types of rounding: round to ∞, round to $-\infty$ and round to 0.

Let $\mathbf{F}$ be the set of all floating point numbers. Then round to ∞, round to $-\infty$ and round to 0 may be defined as

$$\begin{aligned} \mathrm{r}_{\infty}(x) &= \min\{\, f \in \mathbf{F} : x \le f \,\}, \\ \mathrm{r}_{-\infty}(x) &= \max\{\, f \in \mathbf{F} : f \le x \,\}, \\ \mathrm{r}_{0}(x) &= \mathrm{signum}(x) \cdot \mathrm{r}_{-\infty}(|x|), \end{aligned}$$

respectively. Round to 0 is also called truncating. Note that for $x \in \mathbf{R}$ we have

$$\mathrm{r}_{-\infty}(x) \le x \le \mathrm{r}_{\infty}(x)$$

and similarly

$$\mathrm{r}_{-\infty}(x) \le x^* \le \mathrm{r}_{\infty}(x).$$

Consider the floating point calculation divide 1 by 3 to find an approximation of 1/3. Using base 10 floating point with 6 significant digits we have

$$3.33333 \times 10^{-1} = \mathrm{r}_{-\infty}(1/3) \le 1/3 \le \mathrm{r}_{\infty}(1/3) = 3.33334 \times 10^{-1}.$$

Thus, the true value of 1/3 lies in the interval $[3.33333 \times 10^{-1}, 3.33334 \times 10^{-1}]$. Similar techniques can be used to provide rigorous mathematical bounds for any floating point calculation. Google on `intlab` and `interval arithmetic` for more information.

All computing hardware that conforms to the IEEE 754 standard supports the default rounding method we have denoted by x^* plus these three additional rounding methods; however, most general purpose programming languages such as C and Fortran don't provide a built-in way to access the additional rounding modes. Matlab doesn't either. Fortunately, the default method of rounding x to the nearest representation x^* is preferred for the numerical algorithms we will be developing in this course.

Let us estimate the errors for an addition problem using three-digit decimal normalized floating point arithmetic with the default rounding.

$$6.19 \times 10^2 +^* 5.82 \times 10^2 = (6.19 \times 10^2 + 5.82 \times 10^2)^* = (12.01 \times 10^2)^* = 1.20 \times 10^3.$$

The generated error in this calculation is

$$\begin{aligned} e_{\text{gen}} &= (6.19 \times 10^2 +^* 5.82 \times 10^2) - (6.19 \times 10^2 + 5.82 \times 10^2) \\ &= -0.001 \times 10^3 = -1 \times 10^0. \end{aligned}$$

To estimate the accumulated (or total) error we need to know what the initial and propagated errors are. If $x^* = 6.19 \times 10^2$ then we may infer that $x \in (6.185 \times 10^2, 6.195 \times 10^2)$. Note that by the banker's round that $(6.185 \times 10^2)^* = 6.18 \times 10^2$ and $(6.195 \times 10^2)^* = 6.20 \times 10^2$. Thus we don't include the endpoints in this interval. Similarly, if $y^* = 5.82 \times 10^2$ then we may infer that $y \in [5.815 \times 10^2, 5.825 \times 10^2]$. In this case we include the endpoints in the interval. It follows that the initial errors are bounded as

$$|e_x| < 0.005 \times 10^2 = 5 \times 10^{-1} \quad \text{and} \quad |e_y| \leq 5 \times 10^{-1}.$$

Hence, the propagated error satisfies

$$|e_{\text{prop}}| = |e_x + e_y| < 1 \times 10^0.$$

We conclude that the cumulative error $e_{\text{tot}} = e_{\text{gen}} + e_{\text{prop}}$ satisfies

$$-2 \times 10^0 < e_{\text{tot}} < 0.$$

Note that this guarantees, no matter what the initial values of x and y were, that the cumulative error is negative.

For a second example, let us work a multiplication problem.

$$(3.60 \times 10^3) * (1.01 \times 10^{-1}) = (3.6360 \times 10^2)^* = 3.64 \times 10^2.$$

Hence the relative generated error satisfies

$$0.00110011001 \leq \tilde{e}_{\text{gen}} = \frac{0.004}{3.6360} \leq 0.00110011002.$$

Note that we have rounded the decimal approximation of the upper bound up towards ∞ and the lower bounds down towards $-\infty$ to ensure that the inequality is mathematically correct. We shall continue with this kind of estimates throughout this example. Since $x^* = 3.60 \times 10^3$ we infer that $x \in [3.595 \times 10^3, 3.605 \times 10^3]$. Therefore

$$-0.00139082059 \leq \frac{-0.005}{3.595} \leq \tilde{e}_x \leq \frac{0.005}{3.605} \leq 0.00138696256.$$

Similarly $y^* = 1.01 \times 10^{-1}$ implies $y \in (1.005 \times 10^{-1}, 1.015 \times 10^{-1})$. Therefore

$$-0.00497512438 \leq \frac{-0.005}{1.005} < \tilde{e}_y < \frac{0.005}{1.015} \leq 0.00492610838.$$

Hence, the relative propagated error satisfies

$$-0.00635902547 < \tilde{e}_{\text{prop}} = \tilde{e}_x \tilde{e}_y + \tilde{e}_x + \tilde{e}_y < 0.00631990327.$$

We conclude that the relative cumulative error $\tilde{e}_{\text{tot}} = \tilde{e}_{\text{gen}} \tilde{e}_{\text{prop}} + \tilde{e}_{\text{gen}} + \tilde{e}_{\text{prop}}$ satisfies

$$-0.00526591109 < \tilde{e}_{\text{tot}} < 0.00742696588$$

Notice that simpler estimates on the errors can be made by neglecting the terms $\tilde{e}_x \tilde{e}_y$ and $\tilde{e}_{\text{gen}} \tilde{e}_{\text{prop}}$ as suggested in the textbook. Of course, in this case, the final estimates on the errors are only approximate and don't represent true mathematical bounds.