1 Computer Arithmetic and Errors

How can we define 'error' in a computation? In its simplest form, it is the difference between the exact answer A, say, and the computed answer, $\tilde{A}$. Hence, we can write,

$$\text{ERROR } = \tilde{A} - A.$$

Since we are usually interested in the magnitude or absolute value of the error we can also define

$$\text{ABSOLUTE ERROR } = \left|\tilde{A} - A\right|.$$

In practical calculations, it is important to obtain an upper bound on the error i.e. a number, E, such that,

$$\left|\tilde{A} - A\right| < E.$$

Clearly, we would like E to be small!

In practice we are often more interested in so-called 'relative error' rather than absolute error and we define,

$$\text{RELATIVE ERROR} = \frac{\left|\tilde{A} - A\right|}{|A|}.$$

This is often expressed as a percentage. Hence, an 'error' of 10^{-5} may be a good or bad 'relative error' depending on the answer. For example,

answer	=	1000	error	=	10^{-5}	very good
answer	=	1	error	=	10^{-5}	good
answer	=	10^{-5}	error	=	10^{-5}	very bad

What are the possible sources of error in a computation?

1. Human error
2. Truncation error
3. Rounding error

A typical 'human error' is

- arithmetic error
- programming error

These errors can be very hard to detect unless they give obviously incorrect solutions. In discussing errors, we shall assume that human errors are **not present.**

1.1 Truncation error

A truncation error is present when some **infinite** process is approximated by a **finite** process. For example, consider the Taylor series expansion,

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \cdots$$

If this formula is used to calculate $f = e^{0.1}$ we get:

$$f = 1 + 0.1 + \frac{(0.1)^2}{2!} + \frac{(0.1)^3}{3!} + \cdots$$

Where do we stop the calculation? How many terms do we include? Theoretically the calculation will never stop. There are always more terms to add on. If we do stop after a finite number of terms, we will not get the exact answer. For example, if we take the first five terms as the approximation we get,

$$f = e^{0.1} \simeq 1 + 0.1 + \frac{(0.1)^2}{2!} + \frac{(0.1)^3}{3!} + \frac{(0.1)^4}{4!} = \bar{f} \approx 1.105$$

For this calculation, the truncation error TE (i.e. the sum of the terms that have been *chopped off*) is,

$$\text{TE} \;=\; \bar{f} - f \;=\; -\frac{(0.1)^5}{5!} - \frac{(0.1)^6}{6!} - \cdots$$

The numerical analyst might try and estimate the size of the truncation error, i.e. |TE|. In this example, we can easily get a rough estimate.

$$\begin{aligned}
|\text{TE}| &= \frac{(0.1)^5}{5!}\left(1 + \frac{0.1}{6} + \frac{(0.1)^2}{6\times 7} + \frac{(0.1)^3}{6\times 7\times 8} + \cdots\right) \\
&\leq \frac{(0.1)^5}{5!}\left(1 + 0.1 + \frac{(0.1)^2}{1\times 2} + \frac{(0.1)^3}{1\times 2\times 3} + \cdots\right) \\
&\leq \frac{(0.1)^5}{5!}e^{0.1} \simeq \frac{0.00001}{120} \times 1.105 \approx 10^{-7} \\
&\therefore \text{ the error in truncating to five terms is approximately } 10^{-7} \text{ at } x = 0.1.
\end{aligned}$$

In general it is much harder to estimate the truncation error!

1.2 Rounding error

In order to introduce the idea of a rounding error, consider the calculation of $\bar{f}$ above.

$$\begin{aligned}
1 &= 1 \\
\frac{(0.1)}{1!} &= 0.1 \\
\frac{(0.1)^2}{2!} &= 0.005 \\
\frac{(0.1)^3}{3!} &= 0.000166\dot{6} \\
\frac{(0.1)^4}{4!} &= 0.000004166\dot{6} \\
\text{summing above} &= 1.10517083\dot{3} \;=\; \bar{f}
\end{aligned}$$

The exact answer to the truncated problem, $\bar{f}$, is an infinite string of digits and, as such, is not very useful. Since we know that it is in error in the seventh decimal place we could round it to six or seven decimal places. For example, rounding to six decimal places gives,

$$\bar{f} \simeq 1.105171 = \tilde{f}$$

where the usual rounding process has been adopted ; namely, if the next figure is 0,1,2,3 or 4 round down; 5,6,7,8 or 9 round up. The difference between $\bar{f}$ and $\tilde{f}$

$$\tilde{f} - \bar{f} = 0.000000166\dot{} = \text{RE}$$

is the rounding error RE. Using the usual rounding process (and rounding to six decimal places) the rounding error is always bounded by $\frac{1}{2}10^{-6}$. Thus, in computing the answer,

$$e^{0.1} \approx \tilde{f} = 1.105171$$

two errors are present and we have,

$$\begin{aligned} \text{ERROR} &= \tilde{f} - f = (\tilde{f} - \bar{f}) + (\bar{f} - f) \\ &= \text{RE} + \text{TE} \\ |\text{ERROR}| &\leq |\text{RE}| + |\text{TE}| \\ &\approx \frac{1}{2}10^{-6} + 10^{-7} \approx \frac{1}{2}10^{-6} \end{aligned}$$

Note that in this case the actual error is dominated by ROUNDING.

1.3 Computer arithmetic

Computers allocate a fixed amount of storage to every number they use. Each number is stored as a fixed string of digits. In practice, so-called *floating point* numbers are used. A computer using four digit decimal arithmetic with floating point numbers would store

37.31	as	$(0.3731, 2)$	=	0.3731×10^{2}
0.00004717	as	$(0.4717, -4)$	=	0.4717×10^{-4}
0.03	as	$(0.3000, -1)$	=	0.3×10^{-1}
14.211	as	$(0.1421, 2)$	=	0.1421×10^{2}

The number pair (p, q) is called a floating point number. p is called the MANTISSA (or REAL PART) and q is the CHARACTERISTIC (or INDEX or EXPONENT). The mantissa is always a fixed number of digits and the index must lie in some range. Typically

$$-256 < \text{INDEX} < 256.$$

If the INDEX goes outside that range then we get underflow (less than -256) or overflow (greater than 256). Some computers/systems automatically replace underflow by the special number 0 (zero). Overflow always gives some sort of error.

We note that the mantissa is always of the form $0.\cdots$ and the digit after the decimal point is always non-zero. Thus, in the third example above 0.03 is stored as $(0.3000, -1)$ and not as $(0.0300, 0)$. We also note that there is no representation of zero. A computer normally has some special representation for this number. We further note, as in the fourth example, that the representation may not be exact.

Finally it should be remembered that in practice computers do not use decimal numbers. They actually use binary numbers. There is often some error in converting decimal numbers to or from binary numbers.

Rounding errors are therefore always present since we can never be certain that a computation has been done exactly. For example, a computer working with four digit, decimal, floating point arithmetic with,

$$A = (0.3333, 2)\,,\ B = (0.4625, 3)$$

would compute,

$$\begin{array}{lcll} A+B & \longrightarrow & (+0.4958, 3) & \neq \quad A+B \\ A-B & \longrightarrow & (-0.4292, 3) & \neq \quad A-B \\ A\times B & \longrightarrow & (+0.1542, 5) & \neq \quad A\times B \end{array}$$

and none is exact.

In a more challenging computation like solving $A\underline{x} = \underline{b}$, where A is a 1000×1000 matrix, there are millions of floating point calculations in all of which small errors are present! Can we be certain of the result? How accurate is it? Can we find algorithms that overcome the problem? All these questions are considered by Numerical Analysts.

1.4 Cancellation

In many books 'cancellation' or more precisely, 'cancellation error' is listed as a fourth source of error. This is not strictly a new source of error but rather a consequence of rounding and truncation errors leading to severe loss of accuracy in certain circumstances. Suppose, for example, that we have two numbers that we know really accurately,

$$\begin{array}{lcll} a & = & 0.642136 & \left(\text{accurate to } \frac{1}{2}10^{-6}\right), \\ b & = & 0.642125 & \left(\text{accurate to } \frac{1}{2}10^{-6}\right). \end{array}$$

Then,

$$a - b = 0.000011$$

and this quantity will contain an error bounded by

$$|\text{error in } a| + |\text{error in } b| \leq 1 \times 10^{-6}.$$

The relative error in $a-b$ is about 10% and is therefore unacceptably large when the relative errors in a and b are only 0.0005%. Moreover, if the errors in the data, a and b, were $\frac{1}{2}10^{-4}$ then the answer would be meaningless!

1.5 Examples

Example 1
Using 3 digit floating point arithmetic find the answer of the calculation,

$$M = \frac{a + b * c}{b + c}$$

when $a = 11.13,\ b = 1.247$ and $c = -0.145$. Identify the rounding error at each stage of the calculation and the total effect of rounding error.
Solution
The representation of $a,\ b$, and c as three digit floating point numbers are:

$a := (+0.111, +2)$	rounding error = -0.0003
$b := (+0.125, +1)$	rounding error = +0.003
$c := (-0.145, 0)$	rounding error = 0

Each calculation is performed as a series of single operations each with their own rounding error. Thus we compute:

$$X := b * c$$
$$Y := a + X$$
$$Z := b + c$$
$$M := Y/Z$$

and we obtain,

$X := (-0.181, 0)$	rounding error = +0.00025
$Y := (+0.109, +2)$	rounding error = -0.019
$Z := (+0.111, +1)$	rounding error = +0.005
$M := (+0.982, +1)$	rounding error =+0.00018

Thus the computed answer is 9.82. The exact answer is 9.8812. Hence, the total effect of rounding error, i.e. the computed value minus the exact value, is -0.06.