Example 4 [continued]

Here are my calculated results for I_{15} [you may get different numbers!]:

$$
\begin{array}{ll}
I_0 = 1 - 1/\mathrm{e} = & 0.632120558 \\
I_1 = 1 - I_0 & 0.367879441 \\
I_2 = 1 - 2I_1 & 0.264241117 \\
I_3 = 1 - 3I_2 & 0.207276647 \\
I_4 = & 0.170893411 \\
I_5 = & 0.14553294 \\
I_6 = & 0.12680237 \\
I_7 = & 0.112383496 \\
I_8 = & 0.100932025 \\
I_9 = & 0.091611769 \\
I_{10} = & 0.083882304 \\
I_{11} = & 0.077294656 \\
I_{12} = & 0.072464148 \\
I_{13} = & 0.057966336 \\
I_{14} = & 0.188471296 \\
I_{15} = & -1.82706944 \\
I_{16} = & 30.233\ldots \\
I_{17} = & -512.9\ldots
\end{array}
$$

Oops! Clearly something is going wrong. [Is it clear? Note that as the integrand runs up from 0 when $x = 0$ to 1 when $x = 1$, the area under the graph lies between 0 and 1, so $0 < I_n < 1$ for all $n \geqslant 0$.] *Question:* Did you spot that I_{15} was 'wrong' by your calculator? *And:* Would you have spotted *anything* wrong with, say, I_{12}?

What has gone wrong?

Any error, even in the 10th or 12th decimal place, is multiplied by 1, then by 2, then by 3, ..., then by 15, so altogether by $15! \approx 1.3\times10^{12}$.

The true value is swamped by the error, even though no large numbers have been used, and only a few operations. A computer can—very easily!—go this wrong in well under a microsecond.

Solutions:

(a) Use Maple with huge precision. Ugh. *Exercise:* Use Maple to get the exact answer. Use *evalf* to print it out, and note that the answer is wrong unless you use more decimal places than the default.

(b) Use a different method: (i) power series expansion of the integrand [works well, and is in the end quite similar to (c) below]; (ii) Simpson's Rule [works, but not well].

(c) Make the errors work for you. If we jiggle with the formula, we get $I_{n-1} = (1-I_n)/n$:

$I_{20} \approx 0$	1.0
$I_{19} \approx 0.05$	0.05
$I_{18} \approx 0.05$	0.0026
$I_{17} \approx 0.052777777$	0.00015
$I_{16} \approx 0.055718954$	8×10^{-6}
$I_{15} \approx 0.059017565$	5×10^{-7}

[The RH column is a bound on the error, divided each time by n from the line above.]

Actually, $I_{15} \approx 0.059017540$.

Errors

There are three main sources:

(a) **Rounding errors**

- Your calculator/computer is finite. Eg it may show/use $\pi = 3.141592654$, when actually $\pi = 3.1415926535897\ldots$.
- Numbers used should ideally be correct to last place shown.
- Depends on computer/calculator, 'guard digits', size of number, phase of moon, ...
- **NB** binary computers do not hold decimals exactly, in general.

(b) **Truncation errors**

- We have to stop 'infinite' process after a finite time, eg
$$\sin x \approx x - \tfrac{1}{6}x^3 + \tfrac{1}{120}x^5 - \tfrac{1}{5040}x^7 + \ldots.$$
- Need help from 'pure maths' theory, usually.

(c) **Blunders**

- Program bugs, misunderstandings.
- Mis-typing of data.
- To cure these, need checks, self-correction, etc.
- [So we prefer methods in which blunders are, somehow, made obvious to those in which a number just drops out at the end and we have no way of knowing whether it is right or wrong.]

Errors are:

(a) **Absolute**

- think: ‘number of decimal places’ [dp].

- $\pi = 3.141593$ **to 6 dp**

As, in fact, $\pi = 3.14159265\ldots$, the *error* [approximate value minus exact value] is $0.00000035\ldots$, equivalently, the *correction* [what you have to add to the approximate value to get the exact] is $-0.00000035\ldots$. We rarely know error or correction very accurately; but if we are quoting 6dp, we should have reason to believe that it lies between $\pm\frac{1}{2}\times10^{-6}$.

(b) **Relative**

- think: ‘number of significant figures’ [sf].

- $\pi = 3.141593$ **to 7 sf.**

$$\textit{relative error} = \frac{\textit{approximate} - \textit{exact}}{\textit{exact}}.$$

Again, the actual error is not usually known very accurately, but when you give answers to [say] 7sf, you should have reason to suppose that the 7th digit is correct.

[Some books have slightly different definitions of error and correction. If the differences matter, you are already in trouble.]

If you add or subtract two numbers, then their absolute errors are also added or subtracted. If you add a million numbers, each known to 10dp, then their sum is only guaranteed to 4dp. You *may* be luckier—but bad luck is not unknown.

If you multiply or divide numbers, then the absolute errors are scaled; if we know $\pi = 3.141593$ to 6dp, then we know $100\times\pi = 314.1593$ to 4dp or $0.001\times\pi = 0.003141593$ to 9dp.

If we *average* our million numbers, by adding them up and dividing by a million, then we lose 6dp in the addition, but get them back with the division *even if* we are unlucky; with luck, we do better. Averaging is a number-friendly process.

If you multiply or divide numbers, then their relative errors are approximately added or subtracted. There is no easy rule for relative errors when you add or subtract numbers.

For 'normal' numbers, it doesn't usually much matter whether you think about dp or sf. *But* in science and engineering, we often deal with very large or very small numbers. In such cases, the number of dp is often irrelevant, and what matters is the number of sf in our answers.

Cancellation error

There is one special case where relative errors are very important when you add or subtract numbers. If the answer is much smaller than the numbers we started with, then significant figures have been lost. This is called cancellation error.

Example: 1000001 has 7sf and 999999 has 6sf; but 1000001−999999=2 has only 1sf. We have lost 5sf.

[See also example 2 from the first lecture.]

Example: In maths, we are taught that

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

But if we try to *use* that formula [without the limit], then:

(a) with h large, say $h = 0.1$, there is a large truncation error, and the results are not very accurate.

(b) with h small, say $h = 0.0000000001$, then $f(x+h)$ and $f(x)$ will be very close together, so we get cancellation error, and so a large relative error—the numerator is known to only very few sf—though only a small absolute error. *Then* we divide by h, which scales the absolute error, so that we get a large absolute error as well, and the result is worthless. Ugh!