11 Errors in numerical computing

Usually, when we solve a problem we may use the methods of algebra or calculus, for example, differential equations, integration, matrix algorithms. But a computer is limited to only a few operations: multiplication, subtraction, multiplication, division and comparisons. Hence, numerical analysis can use only those to solve the engineering problems. The main difference between the analytical result and the numerical result, apart from the solution limitations, is that the latter is always an approximation. However, the approximation may be as accurate as needed, accuracy determined by the application.

To store the numbers in computer memory and simplify the operations (addition, multiplication), decimal numbers need to be converted from base 10 (decimal) to base 2 (binary). To give output to the user in decimal notation, the process is reversed and the number is conversed from base 2 to base 10. If a fractional part of a base 10 number is not a finite base 2 expansion, a number cannot be stored in its exact form, part of the number is truncated and lost.

11.1 Kinds of errors in numerical procedures

Different errors may appear in numerical computation. Some of them are a result of how a computer stores data or does arithmetic operations, but other sources may cause problems as well.

Main types of errors in computations are (Gerald and Wheatley 2004):

- errors in data – problems almost always depend on measurements of doubtful accuracy (and containing measurement noise); what is more, the mathematical and physical models may not reflect the situation ideally;
- blunders – human involvement in input preparation or output interpretation may cause blunders and gross errors to occur; double-checking the data and results or executing a test run is a way to reduce the errors (but is not a guarantee to avoid this kind of errors);
- truncation error – truncation error refers to errors caused by the numerical methods themselves; it can be compared to approximating a function with a Taylor series, which limits the number of sum elements (it can not be infinite); the difference between the series approximation and a function is due to the truncation;
- propagated error – error in the succeeding steps of the algorithm caused by an earlier error.
- round-off error – computers represent numbers (excluding integers and some fractions) with some imprecision; floating point numbers have fixed word length (limited number of bytes to store a binary representation of a number in computer memory), thus, the true values are not expressed exactly; if floating-point numbers are rounded (while being stored) the round off error is smaller than if the trailing digits were truncated.

When a computer performs arithmetic operations on floating-point numbers, the results are not exact (unless numbers are powers of 2). Numbers are stored according to a IEEE standard (IEEE 754) with a limited number of bytes to store a number with single, double or extended precision in a form of sign, fraction part and an exponent. Since the fraction part is a sum of particular powers of 2, only a finite number of different values can be stored in a computer (there are gaps between the values), which is a source of a round-off error. Hence, **if a comparison between two floats is needed, never check for their equality** – although the result from a mathematical point of view may seem the same, the representation due to computational error may be different. Instead of `A == B`, check if `A - B < tolerance`, where `tolerance` is a very small number that is acceptable as a difference between the two values.

Round-off errors may appear even when an algorithm used is exact (gives precise value, truncation error is equal to 0). A computational error is the sum of the two: a round-off error and a truncation error. Sometimes, even a small error during a long calculation can be magnified to a large value.

11.2 Anomalies with floating-points

When adding a set of numbers, an order, in which the operations are done, is important. Adding the numbers starting from the smallest in magnitude to the largest gives more accurate result that if they were added from largest to smallest (the magnitude of the number must be kept in memory, so the little details may be lost).

It is important to be aware of the flaws of the floating-point arithmetic. For example, adding 0.001 a hundred times should give 1.0, but it often does not. Moreover, the statements below are not always true:

$$\begin{aligned}(X+Y)+Z &= X+(Y+Z)\\ (X*Y)*Z &= X*(Y*Z)\\ X*(Y+Z) &= X*Y+X*Z\end{aligned}$$

11.3 Exercises

Exercise 11.1. Using the values given in Table 11.1, compute Z:

$$Z=\frac{(X+Y)^2-2XY-Y^2}{X^2}$$

The expression can be reduced to X^2/X^2, so the result in all cases should be equal to 1 (if $X \neq 0$). Use the above version of the equation, to see the impact of mathematical operations on numbers with different magnitude.

Table 11.1. Data for Exercise 11.1

X	Y
0.01	1000
0.001	1000
0.0001	1000
0.00001	1000
0.000001	1000
0.0000001	1000

Exercise 11.2. Let

$$x = 11x - 10x$$

Starting with x:

- $x = 0.5$,
- $x = 0.1$

compute the right-hand side of the equation and use the determined value in the next iteration. Do 20 iterations. What is the final value of x?

Exercise 11.3. Find the Taylor series for $f(x) = \frac{1}{x}$, expanded about the point $x = 2$, using the formula:

$$\frac{1}{x} = \sum_{i=0}^{\infty} (-1)^i 2^{-1-n} (x-2)^i.$$

Write a program that displays the computed value at $x = 2.5$, the absolute error, and the relative error, for:[1]

- a series of three terms;
- a series of four terms;
- a series of five terms;
- repeat points above for $x = 3$.

References

Burden, R. L. and Faires, J. D. (2011). *Numerical Analysis*. Cengage, 9th edition.

Gerald, C. F. and Wheatley, P. O. (2004). *Applied Numerical Analysis*. Pearson Addison Wesley.

Kiusalaas, J. (2013). *Numerical methods in Python*.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical recipes. The art of scientific computing*. Cambridge University Press, Cambridge, 3rd edition.

Sauer, T. (2012). *Numerical Analysis*. Pearson, USA, 2nd edition.

Stoer, J. and Bulirsch, R. (2013). *Introduction to Numerical Analysis*, volume 12. Springer Science & Business Media, 2nd edition.

[1] The exercise adapted from Gerald and Wheatley (2004)