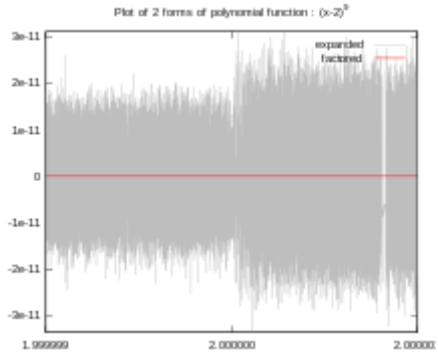
Loss of significance

Loss of significance is an undesirable effect in calculations using finite-precision arithmetic such as floating-point arithmetic. It occurs when an operation on two numbers increases relative error substantially more than it increases absolute error, for example in subtracting two nearly equal numbers (known as **catastrophic cancellation**). The effect is that the number of significant digits in the result is reduced unacceptably. Ways to avoid this effect are studied in numerical analysis.



Example of LOS in case of computing 2 forms of the same function

Contents

Demonstration of the problem

The effect can be demonstrated with decimal numbers. The following example demonstrates loss of significance for a decimal floating-point data type with 10 significant digits:

Consider the decimal number

```
x = 0.1234567891234567890
```

A floating-point representation of this number on a machine that keeps 10 floating-point digits would be

```
y = 0.1234567891
```

which is fairly close when measuring the error as a percentage of the value. It is very different when measured in order of precision. The value 'x' is accurate to 10×10^{-19}, while the value 'y' is only accurate to 10×10^{-10}.

Now perform the calculation

```
x - y = 0.1234567891234567890 − 0.1234567890000000000
```

The answer, accurate to 20 significant digits, is

```
0.0000000001234567890
```

However, on the 10-digit floating-point machine, the calculation yields

```
0.1234567891 − 0.1234567890 = 0.0000000001
```

In both cases the result is accurate to same order of magnitude as the inputs (−20 and −10 respectively). In the second case, the answer seems to have one significant digit, which would amount to loss of significance. However, in computer floating-point arithmetic, all operations can be viewed as being performed on antilogarithms, for which the rules for significant figures indicate that the number of significant figures remains the same as the smallest number of significant figures in the mantissas. The way to indicate this and represent the answer to 10 significant figures is

$$1.000\,000\,000 \times 10^{-10}$$

Workarounds

It is possible to do computations using an exact fractional representation of rational numbers and keep all significant digits, but this is often prohibitively slower than floating-point arithmetic.

One of the most important parts of numerical analysis is to avoid or minimize loss of significance in calculations. If the underlying problem is well-posed, there should be a stable algorithm for solving it.

Sometimes clever algebra tricks can change an expression into a form that circumvents the problem. One such trick is to use the well-known equation

$$(x - y)(x + y) = x^2 - y^2$$

So with the expression $x - y$, multiply numerator and denominator by $x + y$ giving

$$\frac{x^2 - y^2}{x + y}$$

Now, can the expression $x^2 - y^2$ be reduced to eliminate the subtraction? Sometimes it can.

For example, the expression $1 - \sqrt{1 - \delta}$ can suffer loss of significant bits if $|\delta|$ is much smaller than 1. So rewrite the expression as

$$\frac{1 - (1 - \delta)}{1 + \sqrt{1 - \delta}}$$

or

$$\frac{\delta}{1 + \sqrt{1 - \delta}}$$

Loss of significant bits

Let *x* and *y* be positive normalized floating-point numbers.

In the subtraction *x* − *y*, *r* significant bits are lost where

$$q \leq r \leq p,$$

$$2^{-p} \leq 1 - \frac{y}{x} \leq 2^{-q}$$

for some positive integers *p* and *q*.

Instability of the quadratic equation

For example, consider the quadratic equation

$$ax^2 + bx + c = 0,$$

with the two exact solutions:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

This formula may not always produce an accurate result. For example, when c is very small, loss of significance can occur in either of the root calculations, depending on the sign of b.

The case $a = 1$, $b = 200$, $c = -0.000015$ will serve to illustrate the problem:

$$x^2 + 200x - 0.000015 = 0.$$

We have

$$\sqrt{b^2 - 4ac} = \sqrt{200^2 + 4 \times 1 \times 0.000015} = 200.00000015\ldots$$

In real arithmetic, the roots are

$$(-200 - 200.00000015)/2 = -200.000000075,$$
$$(-200 + 200.00000015)/2 = 0.000000075.$$

In 10-digit floating-point arithmetic:

$$(-200 - 200.0000001)/2 = -200.00000005,$$
$$(-200 + 200.0000001)/2 = 0.00000005.$$

Notice that the solution of greater magnitude is accurate to ten digits, but the first nonzero digit of the solution of lesser magnitude is wrong.

Because of the subtraction that occurs in the quadratic equation, it does not constitute a stable algorithm to calculate the two roots.

A better algorithm

A careful floating-point computer implementation combines several strategies to produce a robust result. Assuming that the discriminant $b^2 - 4ac$ is positive, and b is nonzero, the computation would be as follows:[1]

$$x_1 = \frac{-b - \operatorname{sgn}(b)\sqrt{b^2 - 4ac}}{2a},$$
$$x_2 = \frac{2c}{-b - \operatorname{sgn}(b)\sqrt{b^2 - 4ac}} = \frac{c}{ax_1}.$$

Here sgn denotes the sign function, where $\operatorname{sgn}(b)$ is 1 if b is positive, and −1 if b is negative. This avoids cancellation problems between b and the square root of the discriminant by ensuring that only numbers of the same sign are added.

To illustrate the instability of the standard quadratic formula compared to this formula, consider a quadratic equation with roots 1.786737589984535 and $1.149782767465722 \times 10^{-8}$. To 16 significant digits, roughly corresponding to double-precision accuracy on a computer, the monic quadratic equation with these roots may be written as

$$x^2 - 1.786737601482363x + 2.054360090947453 \times 10^{-8} = 0.$$

Using the standard quadratic formula and maintaining 16 significant digits at each step, the standard quadratic formula yields

$$\sqrt{\Delta} = 1.786737578486707,$$
$$x_1 = (1.786737601482363 + 1.786737578486707)/2 = 1.786737589984535,$$
$$x_2 = (1.786737601482363 - 1.786737578486707)/2 = 0.000000011497828.$$

Note how cancellation has resulted in x_2 being computed to only 8 significant digits of accuracy.

The variant formula presented here, however, yields the following:

$$x_1 = (1.786737601482363 + 1.786737578486707)/2 = 1.786737589984535,$$
$$x_2 = 2.054360090947453 \times 10^{-8}/1.786737589984535 = 1.149782767465722 \times 10^{-8}.$$

Note the retention of all significant digits for x_2.

Note that while the above formulation avoids catastrophic cancellation between b and $\sqrt{b^2 - 4ac}$, there remains a form of cancellation between the terms b^2 and $-4ac$ of the discriminant, which can still lead to loss of up to half of correct significant digits.[2][3] The discriminant $b^2 - 4ac$ needs to be computed in arithmetic of twice the precision of the result to avoid this (e.g. quad precision if the final result is to be accurate to full double precision).[4] This can be in the form of a fused multiply-add operation.[2]

To illustrate this, consider the following quadratic equation, adapted from Kahan (2004):[2]

$$94906265.625x^2 - 189812534x + 94906268.375.$$

This equation has $\Delta = 7.5625$ and roots

$$x_1 = 1.000000028975958,$$
$$x_2 = 1.000000000000000.$$

However, when computed using IEEE 754 double-precision arithmetic corresponding to 15 to 17 significant digits of accuracy, Δ is rounded to 0.0, and the computed roots are

$$x_1 = 1.000000014487979,$$
$$x_2 = 1.000000014487979,$$

which are both false after the 8th significant digit. This is despite the fact that superficially, the problem seems to require only 11 significant digits of accuracy for its solution.

See also

- Round-off error
- Kahan summation algorithm
- Karlsruhe Accurate Arithmetic
- Exsecant
- Exponential minus 1
- Natural logarithm plus 1
- Example in wikibooks: Cancellation of significant digits in numerical computations

References

1. Press, William Henry; Flannery, Brian P.; Teukolsky, Saul A.; Vetterling, William T. (1992). "Section 5.6: Quadratic and Cubic Equations" (http://www.nrbook.com/a/bookcpdf.php). *Numerical Recipes in C* (2 ed.).
2. Kahan, William Morton (2004-11-20). "On the Cost of Floating-Point Computation Without Extra-Precise Arithmetic" (http://www.cs.berkeley.edu/~wkahan/Qdrtcs.pdf) (PDF). Retrieved 2012-12-25.
3. Higham, Nicholas John (2002). *Accuracy and Stability of Numerical Algorithms* (2 ed.). SIAM. p. 10. ISBN 978-0-89871-521-7.
4. Hough, David (March 1981). "Applications of the proposed IEEE 754 standard for floating point arithmetic". *Computer*. IEEE. **14** (3): 70–74. doi:10.1109/C-M.1981.220381 (https://doi.org/10.1109%2FC-M.1981.220381).