CS/MATH 3414 — Numerical Methods

Lecture 2
Floating-Point Arithmetic and Error Analysis

Virginia Tech Department of Computer Science, Fall 2026
Instructor: Sumeet Khatri
Date: September 1, 2026

Recap of Lecture 1 on floating-point numbers

- A floating-point number system is given by a fixed set of digits and a fixed range of the exponent when expressed in scientific notation. We distinguish between *normal* and *subnormal* floating-point numbers.
- Floating-point numbers are needed because computer memory is finite.
- The finiteness comes at the cost of *round-off error*. The *ties-to-even* (or *rounds-to-even*) is a common rounding convention used.
- Every floating-point system has its own *unit round-off error* and the so-called *machine epsilon*.
- The current IEEE 754 standards specify floating-point formats such as `binary32` ("single") and `binary64` ("double"), which are almost universally used in computing systems.

Learning Objectives

- State and explain the definitions of floating-point arithmetic operations.
- State and explain catastrophic cancellation and provide methods to remedy it.
- Use numerical experiments to reveal finite-precision effects in floating-point arithmetic.

Primary References

- Burden and Faires, Section 1.2.
- Cheney and Kincaid, Sections 1.1, 1.3, 1.4.
- D. Goldberg, "What Every Computer Scientist Should Know About Floating-Point Arithmetic," *ACM Computing Surveys*, vol. 23, no. 1, pp. 5–48, 1991.

Note: much of the text here comes directly from, or is a summarized form of, the text in the above two references.

1 Floating-point arithmetic operations

In addition to inaccurate representation of numbers arising from the use of a given floating-point number system, the arithmetic performed in a computer is not exact. The arithmetic involves manipulating binary digits by various shifting, or logical, operations. Since the actual mechanics of these operations on an actual machine are not pertinent to this discussion, we shall devise our own approximation to computer arithmetic. Although our arithmetic will not give the exact picture, it suffices to explain the problems that occur. (For an explanation of the manipulations actually involved, the reader may consult texts on computer systems architecture.)

Recall from Lecture 1 that a floating-point number system is defined by the collection $\mathcal{F} \equiv (b, p, k_{\min}, k_{\max})$, where $b \in \{2, 3, \dots\}$ is the *base*, $p \in \{1, 2, \dots\}$ is the *precision*, and $k_{\min}$ and $k_{\max}$ are the minimum and maximum values, respectively, of the exponent, where $k_{\min} < k_{\max}$ and both $k_{\min}, k_{\max}$ can be any integers. Then, a floating point number is:

$$(-1)^s (d_1.d_2 d_3 \cdots d_p)_b \times b^k \quad \text{(normal number)}, \tag{1}$$

$$(-1)^s (0.d_2 d_3 \cdots d_p)_b \times b^{k_{\min}} \quad \text{(subnormal number)}. \tag{2}$$

(For further details, see the Lecture 1 notes.)

Definition 1 (Floating-point arithmetic operations)**.** Consider a floating-point number system $\mathcal{F}$ and its corresponding function $\mathrm{fl}_{\mathcal{F}}(\cdot)$, which converts any given real number into its representation within that system under a given rounding rule (assume ties-to-even). For any two real numbers $x, y \in \mathbb{R}$, we define the floating-point arithmetic operations of addition, subtraction, multiplication, and division as follows:

$$x +_{\mathcal{F}} y \coloneqq \mathrm{fl}_{\mathcal{F}}\big(\mathrm{fl}_{\mathcal{F}}(x) + \mathrm{fl}_{\mathcal{F}}(y)\big), \tag{3}$$

$$x -_{\mathcal{F}} y \coloneqq \mathrm{fl}_{\mathcal{F}}\big(\mathrm{fl}_{\mathcal{F}}(x) - \mathrm{fl}_{\mathcal{F}}(y)\big), \tag{4}$$

$$x \times_{\mathcal{F}} y \coloneqq \mathrm{fl}_{\mathcal{F}}\big(\mathrm{fl}_{\mathcal{F}}(x) \times \mathrm{fl}_{\mathcal{F}}(y)\big), \tag{5}$$

$$x \div_{\mathcal{F}} y \coloneqq \mathrm{fl}_{\mathcal{F}}\big(\mathrm{fl}_{\mathcal{F}}(x) \div \mathrm{fl}_{\mathcal{F}}(y)\big). \tag{6}$$

This arithmetic corresponds to performing exact arithmetic on the floating-point representations of x and y and then converting the exact result to its floating-point representation.

Example 1. Let us consider the base-10 floating number system given by $\mathcal{F} = (10, 5, -4, 5)$.

1. Suppose that $x = \frac{5}{7}$ and $y = \frac{1}{3}$. Use five-digit chopping for calculating $x +_{\mathcal{F}} y$, $x -_{\mathcal{F}} y$, $x \times_{\mathcal{F}} y$, $x \div_{\mathcal{F}} y$.

   Solution: Note that

   $$x = \frac{5}{7} = 0.\overline{714285}, \quad y = \frac{1}{3} = 0.\overline{3}. \tag{7}$$

   Therefore, the five-digit chopping values of x and y are

   $$\mathrm{fl}(x) = 7.1428 \times 10^{-1}, \quad \mathrm{fl}(y) = 3.3333 \times 10^{-1}. \tag{8}$$

Thus,

$$x +_{\mathcal{F}} y = \text{fl}\left(\text{fl}(x) + \text{fl}(y)\right) \tag{9}$$
$$= \text{fl}\left(7.1428 \times 10^{-1} + 3.3333 \times 10^{-1}\right) \tag{10}$$
$$= \text{fl}\left(1.04761 \times 10^{0}\right) \tag{11}$$
$$= 1.0476 \times 10^{0}. \tag{12}$$

The true value is $x + y = \frac{22}{21}$, so we have

$$\text{Absolute error} = \left|\frac{22}{21} - 1.0476 \times 10^{1}\right| = 0.190 \times 10^{-4}, \tag{13}$$
$$\text{Relative error} = \left|\frac{0.190 \times 10^{-4}}{22/21}\right| = 0.182 \times 10^{-4}. \tag{14}$$

The table below lists this and all of the other values.

Operation	Result	Actual value	Absolute error	Relative error
$x +_{\mathcal{F}} y$	1.0476×10^{0}	$22/21$	0.190×10^{-4}	0.182×10^{-4}
$x -_{\mathcal{F}} y$	3.8095×10^{-1}	$8/21$	0.238×10^{-5}	0.625×10^{-5}
$x \times_{\mathcal{F}} y$	2.3809×10^{-1}	$5/21$	0.524×10^{-5}	0.220×10^{-4}
$x \div_{\mathcal{F}} y$	2.1428×10^{0}	$15/7$	0.571×10^{-4}	0.267×10^{-4}

The unit roundoff error for this floating-point system is $\frac{1}{2}b^{-p+1} = \frac{1}{2} \times 10^{-4} = 5 \times 10^{-4}$. So the relative error is less than the unit roundoff, which we consider to mean that the arithmetic produces satisfactory five-digit results. This is not the case in the following example.

2. Now suppose that in addition to x and y we have $u = 0.741251$, $v = 98765.9$, and $w = 0.111111 \times 10^{-4}$. Determine the five-digit chopping values of $x -_{\mathcal{F}} u$, $(x -_{\mathcal{F}} u) \div_{\mathcal{F}} w$, $(x -_{\mathcal{F}} u) \times_{\mathcal{F}} v$, and $u +_{\mathcal{F}} v$.

   Solution: These numbers have been chosen to illustrate some problems that can arise with floating-point arithmetic. Because x and u are nearly the same, their difference is small. The absolute error for $x -_{\mathcal{F}} u$ is

$$|(x - u) - (x -_{\mathcal{F}} u)| = \left|(x - u) - \left(\text{fl}(x)\left(\text{fl}(x) - \text{fl}(u)\right)\right)\right| \tag{15}$$
$$= \left|\left(\frac{5}{7} - 0.714251\right) - \left(\text{fl}\left(0.71428 \times 10^{0} - 0.71425 \times 10^{0}\right)\right)\right| \tag{16}$$
$$= \left|0.347143 \times 10^{-4} - \text{fl}\left(0.00003 \times 10^{0}\right)\right| \tag{17}$$
$$= 0.47143 \times 10^{-5}. \tag{18}$$

   This approximation has a small absolute error but a large relative error:

$$\left|\frac{0.47143 \times 10^{-5}}{0.347143 \times 10^{-4}}\right| \leq 0.136. \tag{19}$$

   The subsequent division by the small number w or multiplication by the large number v magnifies the absolute error without modifying the relative error. The addition of the large and small numbers u and v produces large absolute error but not large relative error. These calculations are shown in the table below.

Operation	Result	Actual value	Absolute error	Relative error
$x -_{\mathcal{F}} u$	0.30000×10^{-4}	0.34714×10^{-4}	0.471×10^{-5}	0.136
$(x -_{\mathcal{F}} u) \div_{\mathcal{F}} w$	2.7000×10^{0}	0.31242×10^{1}	0.424	0.136
$(x -_{\mathcal{F}} u) \times_{\mathcal{F}} v$	2.9629×10^{0}	0.34285×10^{1}	0.465	0.136
$u +_{\mathcal{F}} v$	0.98765×10^{5}	0.98766×10^{5}	0.161×10^{1}	0.163×10^{-4}

Example 2. If x, y, and z are machine numbers in a 32-bit computer, then what upper bound can be given for the relative round-off error in computing $z(x + y)$?

Solution: In the computer, the calculation of $x + y$ would be done first. This arithmetic operation produces the machine number $\text{fl}(x + y)$, which differs from $x + y$ due to roundoff. From Lecture 1, we know that there is a δ_1 such that

$$\text{fl}(x + y) = (x + y)(1 + \delta_1) \quad |\delta_1| \leq 2^{-24}. \tag{20}$$

When z multiplies the machine number $\text{fl}(x + y)$, the result is the machine number $\text{fl}\left(z\,\text{fl}(x + y)\right)$, because z is already a machine number. This, too, differs from its exact counterpart, and we have that

$$\text{fl}\left(z\,\text{fl}(x + y)\right) = z\,\text{fl}(x + y)(1 + \delta_2), \quad |\delta_2| \leq 2^{-24}. \tag{21}$$

Putting both equations together, we have

$$\text{fl}\left(z\,\text{fl}(x + y)\right) = z(x + y)(1 + \delta_1)(1 + \delta_2) \tag{22}$$

$$= z(x + y)(1 + \delta'), \tag{23}$$

where $\delta' = \delta_1 + \delta_2 + \delta_1\delta_2$. Then, using the triangle inequality, we obtain

$$|\delta'| = |\delta_1 + \delta_2 + \delta_1\delta_2| \leq |\delta_1| + |\delta_2| + |\delta_1\delta_2| \leq 2 \times 2^{-24} + 2^{-48} = 2^{-23} + 2^{-48}. \tag{24}$$

Later in this lecture, we will see a more general error analysis of multiplication.

2 Pitfalls and how to avoid them

2.1 Loss of significance

In Example 1, we saw that substracting two numbers that are very close to each other can result in large relative error...in fact, this is one of the most common causes for a deterioration in significant digits. This effect is potentially quite serious and can be catastrophic. The closer these two numbers are to each other, the more pronounced is the effect....

Suppose two nearly equal numbers, x and y, with $x > y$, have the k-digit decimal representations

$$\text{fl}(x) = 0.d_1d_2 \cdots d_p\alpha_{p+1}\alpha_{p+2} \cdots \alpha_k \times 10^n, \tag{25}$$

$$\text{fl}(y) = 0.d_1d_2 \cdots d_p\beta_{p+1}\beta_{p+2} \cdots \beta_k \times 10^n. \tag{26}$$

The floating-point form of $x - y$ is

$$\text{fl}(\text{fl}(x) - \text{fl}(y)) = 0.\sigma_{p+1}\sigma_{p+2} \cdots \sigma_k \times 10^{n-p}, \tag{27}$$

where

$$0.\sigma_{p+1}\sigma_{p+2}\cdots\sigma_k = 0.\alpha_{p+1}\alpha_{p+2}\cdots\alpha_k - 0.\beta_{p+1}\beta_{p+2}\cdots\beta_k. \tag{28}$$

The floating-point number used to represent $x - y$ has at most $k - p$ digits of significance. However, in most calculation devices, $x - y$ will be assigned k digits, with the last p being either zero or randomly assigned. Any further calculations involving $x - y$ retain the problem of having only $k - p$ significant digits, since a chain of calculations is no more accurate than its weakest portion.

If a finite-digit representation or calculation introduces an error, further enlargement of the error occurs when dividing by a number with small magnitude (or, equivalently, when multiplying by a number with large magnitude). Suppose, for example, that the number z has the finite-digit approximation $z + \delta$, where the error δ is introduced by representation or by previous calculation. Now, divide by $\varepsilon = 10^{-n}$, where $n > 0$. Then,

$$\frac{z}{\varepsilon} \approx \mathrm{fl}\left(\frac{\mathrm{fl}(z)}{\mathrm{fl}(\varepsilon)}\right) = (z + \delta) \times 10^n. \tag{29}$$

The absolute error in this approximation, $|\delta| \times 10^n$, is the original absolute error, $|\delta|$, multiplied by the factor 10^n!

Definition 2 (Correct significant digits)**.** Let $x \neq 0$, let $\hat{x}$ approximate x, and let $b \in \{2, 3, \dots\}$ be the base. We say that $\hat{x}$ has at least t correct base-b significant digits if the relative error is less than the unit roundoff value for a base-b floating-point system with t digits of precision:

$$\frac{|\hat{x} - x|}{|x|} \leq \frac{1}{2} b^{1-t}. \tag{30}$$

Example 3. Let $x = 0.54617$ and $y = 0.54601$. Use four-digit arithmetic to approximate $p - q$ and determine the absolute and relative errors using: (a) rounding; (b) chopping.

Solution: The exact value of $z = x - y$ is $z = 0.00016$.

(a) Suppose that the subtraction is performed using four-digit rounding arithmetic. Rounding p and q to four digits gives $\hat{p} = 0.5462$ and $\hat{q} = 0.5460$, respectively, and $\hat{r} = \hat{p} - \hat{q} = 0.0002$ is the four-digit approximation to r. Since

$$\frac{|r - \hat{r}|}{|r|} = \frac{|0.00016 - 0.0002|}{|0.00016|} = 0.25, \tag{31}$$

the result has only one significant digit, whereas $\hat{p}$ and $\hat{q}$ were accurate to four and five significant digits, respectively.

(b) If chopping is used to obtain the four digits, then the four-digit approximations to p and q and r are $\hat{p} = 0.5461$, $\hat{q} = 0.5460$, and $\hat{r} = \hat{p} - \hat{q} = 0.0001$. This gives

$$\frac{|r - \hat{r}|}{|r|} = \frac{|0.00016 - 0.0001|}{|0.00016|} = 0.375, \tag{32}$$

which also results in only one significant digit.

Example 4. If $x = 0.3721448693$ and $y = 0.3720214371$, then what is the relative error in the computation of $x - y$ in a computer that has five decimal digits of precision?

Solution: The numbers would first be rounded to $\hat{x} = 0.37214$ and $\hat{y} = 0.37202$. Then, we have $\hat{x} - \hat{y} = 0.00012$, while the correct answer is $x - y = 0.0001234322$. The relative error involved is

$$\frac{|(x-y) - (\hat{x} - \hat{y})|}{|x - y|} = \frac{0.0000034322}{0.0001234322} \approx 3 \times 10^{-2}. \tag{33}$$

This magnitude of relative error must be judged quite large when compared with the relative errors of $\hat{x}$ and $\hat{y}$. (We know they should not exceed the unit roundoff of $\frac{1}{2} \times 10^{-4}$, and in this case they are approximately 1.3×10^{-5}.) Indeed, we observe that the computed difference of 0.00012 has only two significant digits, while we would expect the numbers and calculations in the computer to have five significant digits.

One remedy for the loss of significant digits is to anticipate that it may occur and then to reprogram. The simplest technique may be to carry out part of the computation in double or long-double floating-point formats. Consider Example 3, but imagine that the calculations to obtain x, y, and $x - y$ are being done in double precision. Suppose that single-precision arithmetic is used thereafter. In the computer, all ten digits of x, y, and $x - y$ are retained, but at the end $x - y$ is rounded to its five-digit form, which is 0.12343×10^{-3}. This answer has five significant digits, as we would like. Of course, the programmer has to know in advance where double-precision arithmetic may be necessary in the computation. Programming everything in double precision may be wasteful if it is not needed. Of course, there may be such serious cancellation of significant digits that even double precision might not help!

The loss of significant digits can also be remedied by a reformulation of the calculations, as illustrated in the next example.

Example 5. The quadratic formula states that the roots of $ax^2 + bx + c = 0$, when $a \neq 0$, are

$$x_{\pm} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{34}$$

Consider this formula applied to the equation $x^2 + 61.10x + 1 = 0$, whose roots are approximately

$$x_+ = -0.01610723, \quad x_- = -62.08390. \tag{35}$$

We will use four-digit rounding arithemtic in the calculation to determine the root. In this equation, b^2 is much larger than $4ac$, so the numerator in the calculation for x_+ involves the *subtraction* of nearly equal numbers. Because

$$\sqrt{b^2 - 4ac} = \sqrt{(62.10)^2 - (4.000)(1.000)(1.000)} \tag{36}$$

$$= \sqrt{3856.0 - 4.000} \tag{37}$$

$$= \sqrt{3852} \tag{38}$$

$$= 62.06, \tag{39}$$

we have

$$\mathrm{fl}(x_+) = \frac{-62.10 + 62.06}{2.000} = \frac{-0.04000}{2.000} = -0.02000, \tag{40}$$

a poor approximation to $x_+ = -0.01611$, with the large relative error

$$\frac{|-0.01611 + 0.02000|}{|-0.01611|} \approx 2.4 \times 10^{-1}. \tag{41}$$

On the other hand, the calculation for x_- involves the *addition* of the nearly equal numbers $-b$ and $-\sqrt{b^2 - 4ac}$. This presents no problem, because

$$\mathrm{fl}(x_-) = \frac{-62.10 - 62.06}{2.000} = \frac{-124.2}{2.000} = -62.10, \tag{42}$$

which has the small relative error of

$$\frac{|-62.08 + 62.10|}{|-62.08|} \approx 3.2 \times 10^{-4}. \tag{43}$$

To obtain a more accurate four-digit rounding approximation for x_+, we change the form of the quadratic formula by *rationalizing the numerator*:

$$x_+ = \frac{-b + \sqrt{b^2 - 4ac}}{2a}\left(\frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}}\right) = \frac{b^2 - (b^2 - 4ac)}{2a(-b - \sqrt{b^2 - 4ac})}, \tag{44}$$

which simplifies to

$$x_+ = \frac{-2c}{b + \sqrt{b^2 - 4ac}}. \tag{45}$$

Using this formula then gives

$$\mathrm{fl}(x_+) = \frac{-2.000}{62.10 + 62.06} = \frac{-2.000}{124.2} = -0.01610, \tag{46}$$

which has the much smaller relative error of 6.2×10^{-4}.

The rationalization technique can also be applied to give the following alternative quadratic formula for x_-:

$$x_- = \frac{-2c}{b - \sqrt{b^2 - 4ac}}. \tag{47}$$

This is the form to use if b is a negative number. In this example, however, the mistaken use of this formula for x_- would result in not only the subtraction of early equal numbers, but also the division by the small result of this subtraction. The inaccuracy that this combination produces is rather large: noting that

$$\mathrm{fl}(x_-) = \frac{-2c}{b - \sqrt{b^2 - 4ac}} = \frac{-2.000}{62.10 - 62.06} = \frac{-2.000}{0.04000} = -50.00, \tag{48}$$

the relative error here is 1.9×10^{-1}.

Theorem 1 (Relative-error amplification and loss of digits under subtraction)**.** Let $x, y \in \mathbb{R}$ with $x \neq y$, and suppose that

$$\widetilde{x} = x(1 + \delta_x), \qquad \widetilde{y} = y(1 + \delta_y), \tag{49}$$

where

$$|\delta_x| \leq \varepsilon, \qquad |\delta_y| \leq \varepsilon. \tag{50}$$

Let

$$d = x - y \tag{51}$$

and first suppose that the perturbed operands are subtracted exactly:

$$\widetilde{d} = \widetilde{x} - \widetilde{y}. \tag{52}$$

Then

$$\frac{|\widetilde{d} - d|}{|d|} \leq \varepsilon\kappa(x, y), \tag{53}$$

where

$$\kappa(x, y) := \frac{|x| + |y|}{|x - y|}. \tag{54}$$

If the final subtraction is also rounded in a range where the usual relative-error model applies, and

$$\hat{d} = \mathrm{fl}(\widetilde{d}) = \widetilde{d}(1 + \delta_s), \qquad |\delta_s| \leq u, \tag{55}$$

then

$$\frac{|\hat{d} - d|}{|d|} \leq \varepsilon\kappa(x, y) + u\,(1 + \varepsilon\kappa(x, y))\,. \tag{56}$$

Finally, suppose that the operands each have at least p correct base-b significant digits. In that case one may take

$$\varepsilon = \frac{1}{2} b^{1-p}. \tag{57}$$

Define

$$\ell := \left\lceil \log_b\big(\kappa(x, y)\big) \right\rceil. \tag{58}$$

If $p > \ell$, then the exact difference $\widetilde{d}$ of the perturbed operands is guaranteed to have at least $p - \ell$ correct significant digits. Thus subtraction can lose approximately

$$\log_b\big(\kappa(x, y)\big) \tag{59}$$

base-b significant digits.

Remark 2. *By the triangle inequality,*

$$\kappa(x, y) \geq 1. \tag{60}$$

The quantity becomes large precisely when same-sign operands are nearly equal.

Proof. The error in the exact subtraction of the perturbed operands is

$$\widetilde{d} - d = \big[x(1 + \delta_x) - y(1 + \delta_y)\big] - (x - y) \tag{61}$$

$$= x\delta_x - y\delta_y. \tag{62}$$

Therefore,

$$|\widetilde{d} - d| \le |x||\delta_x| + |y||\delta_y| \tag{63}$$
$$\le \varepsilon\big(|x| + |y|\big). \tag{64}$$

Dividing by $|d| = |x - y|$ gives

$$\frac{|\widetilde{d} - d|}{|d|} \le \varepsilon \frac{|x| + |y|}{|x - y|} = \varepsilon\kappa(x, y). \tag{65}$$

To see sharpness when x and y are nonzero, let

$$s = \operatorname{sign}(x - y) \tag{66}$$

and choose

$$\delta_x = \varepsilon s \operatorname{sign}(x), \qquad \delta_y = -\varepsilon s \operatorname{sign}(y). \tag{67}$$

Then

$$x\delta_x - y\delta_y = \varepsilon s\big(|x| + |y|\big), \tag{68}$$

so equality holds in the absolute bound. The cases in which one operand is zero follow by the same one-term construction.

For the rounded subtraction,

$$\hat{d} - d = \widetilde{d}(1 + \delta_s) - d \tag{69}$$
$$= (\widetilde{d} - d) + \delta_s\widetilde{d}. \tag{70}$$

The first part has already been bounded. Moreover,

$$|\widetilde{d}| \le |d| + |\widetilde{d} - d| \tag{71}$$
$$\le |d| \left(1 + \varepsilon\kappa(x, y)\right). \tag{72}$$

Consequently,

$$\frac{|\hat{d} - d|}{|d|} \le \frac{|\widetilde{d} - d|}{|d|} + |\delta_s| \frac{|\widetilde{d}|}{|d|} \tag{73}$$
$$\le \varepsilon\kappa(x, y) + u\left(1 + \varepsilon\kappa(x, y)\right), \tag{74}$$

which proves Equation (56).

For the digit statement, the definition of ℓ implies

$$\kappa(x, y) \le b^{\ell}. \tag{75}$$

Using Equation (53),

$$\frac{|\widetilde{d} - d|}{|d|} \le \frac{1}{2} b^{1-p} b^{\ell} \tag{76}$$
$$= \frac{1}{2} b^{1-(p-\ell)}. \tag{77}$$

By Definition 2, this guarantees at least $p - \ell$ correct base-b significant digits. Sharpness of the amplification factor shows that no uniformly better worst-case digit guarantee is possible without additional structure. ■

Remark 3 (Interpreting Theorem 1)**.** *If x and y have the same sign and are nearly equal, then*

$$\kappa(x, y) = \frac{|x| + |y|}{|x - y|} \approx \frac{2|x|}{|x - y|}. \tag{78}$$

If the difference is about β^{-q} times the size of the operands, then

$$\kappa(x, y) \approx 2\beta^q, \tag{79}$$

and approximately q base-β digits can be lost. In decimal arithmetic, the practical rule is

$$\textit{decimal digits lost} \approx \log_{10}\left(\frac{|x| + |y|}{|x - y|}\right). \tag{80}$$

This is a logarithmic restatement of the condition-number bound, not a claim that exactly that many printed digits always disappear.

If the operands have opposite signs, subtraction is effectively addition of magnitudes. In that case,

$$|x - y| = |x| + |y|, \tag{81}$$

so

$$\kappa(x, y) = 1. \tag{82}$$

There is then no cancellation-induced amplification of relative input errors.

Example 6. Let the exact decimal data be

$$x = 1.23456784, \qquad y = 1.23456711. \tag{83}$$

Their exact difference is

$$d = x - y = 7.3 \times 10^{-7}. \tag{84}$$

Rounding each input to eight significant decimal digits gives

$$\widetilde{x} = 1.2345678, \qquad \widetilde{y} = 1.2345671. \tag{85}$$

The subtraction of the represented values gives

$$\widetilde{d} = \widetilde{x} - \widetilde{y} = 7.0 \times 10^{-7}. \tag{86}$$

The relative error is

$$\frac{|\widetilde{d} - d|}{|d|} = \frac{3.0 \times 10^{-8}}{7.3 \times 10^{-7}} \tag{87}$$

$$\approx 4.11 \times 10^{-2}. \tag{88}$$

Thus, although the operands were stored to eight significant digits, the difference has two correct decimal significant digits according to Definition 2, but not three.
The subtraction condition number is

$$\kappa(x, y) = \frac{1.23456784 + 1.23456711}{7.3 \times 10^{-7}} \tag{89}$$

$$\approx 3.3824 \times 10^6. \tag{90}$$

Hence,

$$\log_{10}\big(\kappa(x, y)\big) \approx 6.53. \tag{91}$$

The condition-number estimate therefore predicts a loss of roughly six to seven decimal digits. The ceiling in Theorem 1 gives a worst-case guarantee of one retained digit; the particular rounded data in this example retain two.

2.2 Avoiding catastrophic cancellation in subtraction

Reformulate the expression before rounding. The most effective remedy is often to use an exact identity that exposes the small quantity directly rather than producing it as the difference of two rounded large quantities.

For non-negative a and b, with $\sqrt{a} + \sqrt{b} \neq 0$, rationalization gives

$$\sqrt{a} - \sqrt{b} = \frac{a - b}{\sqrt{a} + \sqrt{b}}. \tag{92}$$

The important point is not merely that the two formulas are algebraically equal. The second formula is useful when $a - b$ is available accurately from the original data. For example,

$$\sqrt{1 + x} - 1 = \frac{x}{\sqrt{1 + x} + 1} \tag{93}$$

uses the original small input x instead of reconstructing it by subtracting two quantities close to one. Section 3.4 analyzes this example in detail.

Other useful identities include

$$x^2 - y^2 = (x - y)(x + y), \tag{94}$$

$$1 - \cos(x) = 2 \sin^2\left(\frac{x}{2}\right), \tag{95}$$

$$\cos(x) - \cos(y) = -2 \sin\left(\frac{x + y}{2}\right) \sin\left(\frac{x - y}{2}\right). \tag{96}$$

For small x, Equation (95) avoids subtracting $\cos(x)$, which is close to one, from one. For nearby x and y, Equation (96) makes the small factor $x - y$ enter through a sine of a small argument rather than through the subtraction of two nearly equal cosine values.

An algebraic reformulation must itself be checked for new numerical hazards. Equation (92), for example, is attractive because the denominator is a sum of nonnegative quantities, but a different rationalization could introduce division by a small number or overflow in an intermediate product. Numerical equivalence must be analyzed, not assumed from symbolic equivalence.

Nested arithmetic and Horner's rule. A polynomial written as a sum of separately evaluated monomials can require many operations and can create cancellation among independently rounded large terms. Let

$$p(x) = \sum_{j=0}^{n} a_j x^j. \tag{97}$$

Horner's rule rewrites the same polynomial in nested form:

$$p(x) = a_0 + x\,(a_1 + x\,(a_2 + \cdots + x(a_{n-1} + xa_n)))\,. \tag{98}$$

Equivalently, define

$$b_n = a_n, \qquad b_k = a_k + xb_{k+1}, \qquad k = n-1, n-2, \ldots, 0. \tag{99}$$

Then $p(x) = b_0$. This form uses n multiplications and n additions, avoids the explicit formation of powers x^j, and often prevents large monomial terms from being rounded independently before they are combined.

3 Direct/forward and backward error analyses

In the error analysis of floating-point arithmetic operations, a **direct error analysis** attempts to determine how computed answers differ from exact answers based on the same data. In contrast, a **backward error analysis** attempts to determine what perturbation of the original data would cause the computer results to be the exact results for a perturbed problem.

3.1 Direct error analysis

Definition 3 (Forward error)**.** Let

$$y = f(a) \tag{100}$$

be the exact answer and let $\hat{y}$ be the computed answer. The forward errors are just the absolute and relative errors we have seen before:

$$\varepsilon_{\text{fwd,abs}} = |\hat{y} - y| \tag{101}$$

and

$$\varepsilon_{\text{fwd,rel}} = \frac{|\hat{y} - y|}{|y|}, \tag{102}$$

for $y \neq 0$.

Remark 4. *If the exact answer is zero, relative forward error is undefined. One must then use an absolute error, a problem-dependent scaling, or a mixed absolute-relative measure such as*

$$\frac{|\hat{y} - y|}{\alpha + |y|}, \tag{103}$$

where $\alpha > 0$ is an appropriate scale.

A direct error analysis proceeds through the actual arithmetic sequence. At every operation, one inserts a local rounding factor or additive local error, and then algebraically propagates those perturbations to the output.

Suppose $x, y, z \in \mathcal{F}$, and consider left-associated summation:

$$\hat{s} = \mathrm{fl}\big(\mathrm{fl}(x + y) + z\big). \tag{104}$$

Introduce one local error for each addition:

$$\mathrm{fl}(x + y) = (x + y)(1 + \delta_1), \qquad |\delta_1| \leq u, \tag{105}$$

and

$$\hat{s} = ((x + y)(1 + \delta_1) + z)\,(1 + \delta_2), \qquad |\delta_2| \le u. \tag{106}$$

Expanding gives

$$\hat{s} = x + y + z + (x + y)\delta_1 + (x + y + z)\delta_2 \tag{107}$$

$$+ (x + y)\delta_1\delta_2. \tag{108}$$

Therefore, the forward error is

$$\hat{s} - (x + y + z) = (x + y)\delta_1 + (x + y + z)\delta_2 + (x + y)\delta_1\delta_2. \tag{109}$$

Taking absolute values yields the direct bound

$$|\hat{s} - (x + y + z)| \le u|x + y| + u|x + y + z| + u^2|x + y|. \tag{110}$$

This analysis is *direct* because it follows the local errors through the computation. Its output is a bound on the forward error.

Products of local rounding factors. A direct rounding-error analysis frequently produces a product of local relative-error factors. For example, after several rounded operations one may arrive at an expression of the form

$$\hat{q} = q \prod_{i=1}^{n} (1 + \delta_i), \qquad |\delta_i| \le u. \tag{111}$$

Each factor $1 + \delta_i$ represents the effect of one local rounding step. It is inconvenient to carry all of the individual quantities δ_i through a long derivation, so one collects the entire product into a single factor $1 + \theta_n$. The following definition and lemma make this notation precise.

Theorem 5 (Accumulation of multiplicative errors)**.** Let $\delta_1, \dots, \delta_n \in \mathbb{R}$ satisfy

$$|\delta_i| \le u, \qquad 1 \le i \le n, \tag{112}$$

and suppose that $nu < 1$. Define

$$\theta_n := \prod_{i=1}^{n} (1 + \delta_i) - 1. \tag{113}$$

Then

$$\prod_{i=1}^{n} (1 + \delta_i) = 1 + \theta_n, \tag{114}$$

and

$$|\theta_n| \le \gamma_n = \frac{nu}{1 - nu}. \tag{115}$$

Proof. By the definition of θ_n, the identity

$$\prod_{i=1}^{n} (1 + \delta_i) = 1 + \theta_n \tag{116}$$

is immediate. The substantive part of the lemma is the bound on θ_n.

Expanding the product and removing its constant term gives

$$\theta_n = \sum_{\emptyset \neq S \subseteq \{1,\dots,n\}} \prod_{i \in S} \delta_i. \tag{117}$$

Here the sum is over all nonempty subsets S of $\{1, \dots, n\}$. Thus the expansion contains all first-order terms δ_i, all second-order terms $\delta_i \delta_j$, and so forth, up to the n-th-order term $\delta_1 \delta_2 \cdots \delta_n$.

Taking absolute values and applying the triangle inequality yields

$$|\theta_n| \leq \sum_{\emptyset \neq S \subseteq \{1,\dots,n\}} \prod_{i \in S} |\delta_i| \tag{118}$$

$$= \prod_{i=1}^{n} (1 + |\delta_i|) - 1 \tag{119}$$

$$\leq (1 + u)^n - 1. \tag{120}$$

The equality in the second line follows by expanding the product $\prod_{i=1}^{n}(1 + |\delta_i|)$: it produces exactly the same subset terms, now with all signs positive.

It remains to obtain a convenient bound for $(1 + u)^n$. The binomial theorem gives

$$(1 + u)^n = \sum_{k=0}^{n} \binom{n}{k} u^k. \tag{121}$$

For every k, one has

$$\binom{n}{k} \leq n^k. \tag{122}$$

Consequently,

$$(1 + u)^n \leq \sum_{k=0}^{n} n^k u^k \tag{123}$$

$$= \sum_{k=0}^{n} (nu)^k \tag{124}$$

$$\leq \sum_{k=0}^{\infty} (nu)^k. \tag{125}$$

Because $nu < 1$, the final series is a convergent geometric series, and therefore

$$\sum_{k=0}^{\infty} (nu)^k = \frac{1}{1 - nu}. \tag{126}$$

Combining the preceding inequalities gives

$$|\theta_n| \leq (1 + u)^n - 1 \tag{127}$$

$$\leq \frac{1}{1 - nu} - 1 \tag{128}$$

$$= \frac{nu}{1 - nu} \tag{129}$$

$$= \gamma_n. \tag{130}$$

This proves the claimed bound. ∎

3.2 Backward error analysis

Definition 4 (Absolute backward error)**.** For the problem $\boldsymbol{y} = f(\boldsymbol{a})$ and computed answer $\hat{\boldsymbol{y}}$, the absolute normwise backward error is

$$\eta_{\text{abs}}(\boldsymbol{a}, \hat{\boldsymbol{y}}) = \inf_{\widetilde{\boldsymbol{a}}} \left\{ \|\widetilde{\boldsymbol{a}} - \boldsymbol{a}\| : f(\widetilde{\boldsymbol{a}}) = \hat{\boldsymbol{y}} \right\}. \tag{131}$$

Thus, backward error asks for the smallest perturbation $\Delta\boldsymbol{a}$ such that

$$\hat{\boldsymbol{y}} = f(\boldsymbol{a} + \Delta\boldsymbol{a}). \tag{132}$$

Definition 5 (Relative backward error)**.** When $\boldsymbol{a} \neq \boldsymbol{0}$, define

$$\eta_{\text{rel}}(\boldsymbol{a}, \hat{\boldsymbol{y}}) = \inf_{\Delta\boldsymbol{a}} \left\{ \frac{\|\Delta\boldsymbol{a}\|}{\|\boldsymbol{a}\|} : f(\boldsymbol{a} + \Delta\boldsymbol{a}) = \hat{\boldsymbol{y}} \right\}. \tag{133}$$

The infimum is needed because the perturbed input may not be unique. It is also possible that $\hat{\boldsymbol{y}}$ does not lie in the range of f, in which case an exact backward interpretation under the chosen perturbation model may not exist.

Example 7 (Forward vs. backward error)**.** Consider decimal arithmetic with three significant digits. Then

$$u = \frac{1}{2} 10^{1-3} = 0.005. \tag{134}$$

Let

$$x = 1.23, \qquad y = 4.56, \tag{135}$$

which are already represented. The exact product is

$$z = xy = 5.6088. \tag{136}$$

The computed product is

$$\hat{z} = \text{fl}(xy) = 5.61. \tag{137}$$

The absolute forward error is

$$|\hat{z} - z| = 0.0012, \tag{138}$$

and the relative forward error is

$$\frac{|\hat{z} - z|}{|z|} = \frac{0.0012}{5.6088} \tag{139}$$

$$\approx 2.14 \times 10^{-4}. \tag{140}$$

Now, in a backward error analysis, we seek a nearby $\widetilde{x}$ such that

$$\hat{z} = \widetilde{x} y. \tag{141}$$

Choose

$$\widetilde{x} = \frac{\hat{z}}{y} \tag{142}$$

$$= \frac{5.61}{4.56} \tag{143}$$

$$\approx 1.23026316. \tag{144}$$

Then

$$\hat{z} = \widetilde{x}y \tag{145}$$

exactly, and

$$\frac{|\widetilde{x} - x|}{|x|} \approx 2.14 \times 10^{-4}. \tag{146}$$

Equivalently, from

$$\hat{z} = xy(1 + \delta), \qquad |\delta| \leq u, \tag{147}$$

one can set

$$\widetilde{x} = x(1 + \delta). \tag{148}$$

The computed product is therefore the exact product of a slightly perturbed input and the unperturbed second input.

3.3 Conditioning

Backward error describes how much the data must be changed to explain a computed result. Conditioning describes how much the exact answer changes when the data are changed.

Definition 6 (Absolute condition number)**.** The local absolute condition number of f at $\boldsymbol{a}$ is

$$\kappa_{\text{abs}}(f, \boldsymbol{a}) = \lim_{\epsilon \to 0^+} \sup_{0 < \|\Delta \boldsymbol{a}\| \leq \epsilon} \frac{\|f(\boldsymbol{a} + \Delta \boldsymbol{a}) - f(\boldsymbol{a})\|}{\|\Delta \boldsymbol{a}\|}. \tag{149}$$

If f is differentiable, then

$$\kappa_{\text{abs}}(f, \boldsymbol{a}) = \|Df(\boldsymbol{a})\|, \tag{150}$$

where $Df(\boldsymbol{a})$ is the derivative or Jacobian and the matrix norm is induced by the selected vector norms.

Definition 7 (Relative condition number)**.** When $\boldsymbol{a} \neq \boldsymbol{0}$ and $f(\boldsymbol{a}) \neq \boldsymbol{0}$, the local normwise relative condition number is

$$\kappa_{\text{rel}}(f, \boldsymbol{a}) = \lim_{\epsilon \to 0^+} \sup_{0 < \|\Delta \boldsymbol{a}\| \leq \epsilon \|\boldsymbol{a}\|} \frac{\|f(\boldsymbol{a} + \Delta \boldsymbol{a}) - f(\boldsymbol{a})\| / \|f(\boldsymbol{a})\|}{\|\Delta \boldsymbol{a}\| / \|\boldsymbol{a}\|}. \tag{151}$$

For differentiable f, this becomes

$$\kappa_{\text{rel}}(f, \boldsymbol{a}) = \frac{\|Df(\boldsymbol{a})\| \|\boldsymbol{a}\|}{\|f(\boldsymbol{a})\|}. \tag{152}$$

For a scalar function $f : \mathbb{R} \to \mathbb{R}$, the familiar formula is

$$\kappa_{\text{rel}}(f, a) = \left| \frac{a f'(a)}{f(a)} \right|. \tag{153}$$

3.4 Detailed example: cancellation in an algebraically unstable formula

Consider

$$f(x) = \sqrt{1 + x} - 1 \tag{154}$$

for small positive x. Rationalization gives the mathematically equivalent expression

$$f(x) = \frac{x}{\sqrt{1 + x} + 1}. \tag{155}$$

The two formulas are identical in exact arithmetic but can behave very differently in floating-point arithmetic.

Assume decimal arithmetic with six significant digits and round to nearest. Let

$$x = 10^{-5}. \tag{156}$$

The exact result is approximately

$$f(x) = 4.9999875000625 \times 10^{-6}. \tag{157}$$

Algorithm A: direct subtraction. The direct formula computes

$$\hat{y}_A = \text{fl}\left(\text{fl}\left(\sqrt{\text{fl}(1 + x)}\right) - 1\right). \tag{158}$$

First,

$$\text{fl}(1 + x) = 1.00001. \tag{159}$$

Next,

$$\sqrt{1.00001} \approx 1.0000049999875. \tag{160}$$

Rounded to six significant digits, this becomes

$$\text{fl}\left(\sqrt{1.00001}\right) = 1.00000. \tag{161}$$

Therefore,

$$\hat{y}_A = 1.00000 - 1.00000 = 0. \tag{162}$$

The relative forward error is

$$\frac{|\hat{y}_A - f(x)|}{|f(x)|} = 1. \tag{163}$$

To find the backward error, solve

$$f(\widetilde{x}) = 0. \tag{164}$$

Because

$$\sqrt{1 + \widetilde{x}} - 1 = 0 \tag{165}$$

implies

$$\widetilde{x} = 0, \tag{166}$$

the relative backward error is

$$\frac{|\widetilde{x} - x|}{|x|} = 1. \tag{167}$$

The direct-subtraction algorithm is not backward stable for this input.

Algorithm B: rationalized expression. The rationalized algorithm computes

$$\hat{y}_B = \mathrm{fl}\left(\frac{x}{\mathrm{fl}\left(\mathrm{fl}(\sqrt{1+x}) + 1\right)} \right). \tag{168}$$

Using the same rounded square root,

$$\mathrm{fl}(\sqrt{1+x}) = 1.00000, \tag{169}$$

so the denominator is

$$\mathrm{fl}(1.00000 + 1) = 2.00000. \tag{170}$$

Thus,

$$\hat{y}_B = 5.00000 \times 10^{-6}. \tag{171}$$

The relative forward error is approximately

$$\frac{|\hat{y}_B - f(x)|}{|f(x)|} \approx 2.5 \times 10^{-6}. \tag{172}$$

For the backward error, solve

$$\sqrt{1+\widetilde{x}} - 1 = \hat{y}_B. \tag{173}$$

This gives

$$\widetilde{x} = (1 + \hat{y}_B)^2 - 1 \tag{174}$$

$$= 2\hat{y}_B + \hat{y}_B^2 \tag{175}$$

$$= 1.0000025 \times 10^{-5}. \tag{176}$$

Therefore,

$$\frac{|\widetilde{x} - x|}{|x|} = 2.5 \times 10^{-6}. \tag{177}$$

The rationalized algorithm has both a small forward error and a small backward error.

Conditioning of the mathematical problem. For

$$f(x) = \sqrt{1+x} - 1, \tag{178}$$

one has

$$f'(x) = \frac{1}{2\sqrt{1+x}}. \tag{179}$$

The relative condition number is

$$\kappa_{\text{rel}}(f,x) = \left|\frac{xf'(x)}{f(x)}\right| \tag{180}$$

$$= \frac{x}{2\sqrt{1+x}\left(\sqrt{1+x}-1\right)} \tag{181}$$

$$= \frac{\sqrt{1+x}+1}{2\sqrt{1+x}}. \tag{182}$$

As $x \to 0^+$,

$$\kappa_{\text{rel}}(f,x) \longrightarrow 1. \tag{183}$$

Thus, the mathematical problem is well-conditioned for small positive x. The failure of Algorithm A is algorithmic instability, not ill-conditioning.