we can now define storage error more precisely

$$\text{Storage error} = \frac{fl(x) - x}{x}$$

= relative error in the exact value and its floating point representation

As you might expect, this is smaller when numbers are rounded than when numbers are chopped.

Chopping $\quad 0 \leq \text{storage error} < \beta^{-t}$

Rounding $\quad -\frac{1}{2}\beta^{-t} \leq \text{storage error} \leq \frac{1}{2}\beta^{-t}$

This means that the storage error is at most 1 digit in the last place of the mantissa

Basic Laws of Algebra (field postulates)

As seen in Example 5, the order that a series of calculations are done can affect the results.

$$\frac{(a \cdot b)}{c} \neq \left(\frac{a}{c}\right) \cdot b$$

However we hope that

$$\frac{(a \cdot b)}{c} = \left(\frac{a}{c}\right) \cdot b + \text{small error}.$$

→ here, small error means 1 digit in the last place.

This is true for most basic algebra rules

Always True

$a+b = b+a$
$a+0 = a$
$a \cdot 0 = 0$
$a \cdot 1 = a$
$a \cdot b = b \cdot a$

True within a small error

$(a+b)+c = a+(b+c)$
$(a \cdot b) \cdot c = a \cdot (b \cdot c)$

True within small error (most of the time)

$a(b+c) = ab+ac$

but this can have a large error if $b \approx -c$

- Modern computers are designed so that any single calculation is accurate to within 1 digit in the last place.

- Mathematically, we can let

  a = some number, $fl(a)$ = stored version of a

  b = some number, $fl(b)$ = stored version of b

- When we say

  $c = a + b$ (or $a - b$, $a * b$, a / b)

  What we actually compute is

  $$fl(c) = fl(\underbrace{fl(a)}_{\text{storage error for } a} + \underbrace{fl(b)}_{\text{storage error for } b})$$

  $\underbrace{\qquad\qquad}$ storage error for the sum of $fl(a)$ and $fl(b)$

- Problems can arise when errors are accumulated over many calculations

- If you assume that a and b are each accurate to within 1 digit, then the computed value of c ($fl(c)$) will be accurate to 1 digit, however

  $$d = a + b + c$$

  can have an error as large as 2 digits in the last place

  $$d = \underbrace{\boxed{a+b}}_{\text{1 digit error}} + c$$

  $\underbrace{\qquad\qquad}$ 2 digits.

- Essentially, each operation has the potential to increase the error by 1 more digit. We will come back to this later.

Conclusion: To obtain good accuracy, use lots of digits

Catastrophic cancellation

Another error that occurs in floating point computing is catastrophic cancellation. This refers to the loss of precision that occurs when you compute

$$x - y \quad \text{and} \quad x + y$$

$$x \approx y \quad \text{and} \quad x \approx -y$$

Example 11

Suppose we are using 4 digits with rounding and

$$x = 1.437 \times 10^4$$

$$y = 1.436 \times 10^4$$

$$x - y = 0.001 \times 10^4 = 1.000 \times 10^1$$

→ 3 digits of precision are lost

This type of error is more subtle to detect and account for.

Example 12

Solve $x^2 - 26x + 1$ using 5 digits with rounding

exact answers

$$x_1 = 13 + \sqrt{168} = 25.9614814\ldots$$

$$x_2 = 13 - \sqrt{168} = 0.038518603\ldots$$

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad b^2 - 4ac = (26)^2 - 4(1)(1) = 676 - 4 = 672$$

$$= \frac{26 + \sqrt{672}}{2(1)} \qquad \sqrt{672} = 25.92296\ldots \xrightarrow{\text{Round}} = 25.923$$

$$x_1 = \frac{26 + 25.923}{2} = \frac{51.923}{2} = 25.9615 \xrightarrow{\text{Round}} 25.961$$

Relative error

$$\text{Rel}\, x_1 = \frac{25.961 - (13+\sqrt{168})}{13+\sqrt{168}} = -1.8542 \times 10^{-5}$$

↑ Very good. 4 digits of accuracy on a computer that uses 5 digits

$$x_2 = \frac{-b - \sqrt{b^2-4ac}}{2a}$$

$$= \frac{26 - 25.923}{2} = \frac{0.077}{2} = 0.0385 \leftarrow \text{only 3 digits}$$

Relative error

$$\text{Rel}\, x_2 = \frac{0.0385 - (13-\sqrt{168})}{(13-\sqrt{168})} = -4.829 \times 10^{-3}$$

due to catestrophic cancellation, we lost 2 digits

Most of the time we can use algebraic manipulation to reduce the effects of catestrophic cancellation

$$x_2 = \frac{-b - \sqrt{b^2-4ac}}{2a} = \frac{-b - \sqrt{b^2-4ac}}{2a} \cdot \left(\frac{-b + \sqrt{b^2-4ac}}{-b + \sqrt{b^2-4ac}} \right)$$

$$= \frac{b^2 - (b^2-4ac)}{2a(-b+\sqrt{b^2-4ac})}$$

$$= \frac{4ac}{2a(-b+\sqrt{b^2-4ac})}$$

$$= \frac{2c}{-b+\sqrt{b^2-4ac}}$$

$$x_2 = \frac{2(1)}{26+25.923} = \frac{2}{51.923} = 0.038518585\ldots \xrightarrow{\text{Round}} = 0.038519$$

Relative error

$$Rel_{x_2} = \frac{0.038519 - (13 - \sqrt{168})}{(13 - \sqrt{168})} = 1.0301 \times 10^{-5}$$

↑ accuracy is now the same size as for x_1

we won't look at this in any more detail than this, but it is an important concept to be aware of.

Real computers

we have been demonstrating floating point computation using fictional computers so that the number of digits are easy to work with. What about real computers? What are their parameters?

Example 13

Hand calculator

$\beta = 10$
$t = 10$ or 12
$e = -99$
$E = 99$

Most computers follow the IEEE specifications for floating point computation (cray supercomputers are the notable exception)

IEEE single precision

$\beta = 2$
$t = 23$
$e = -126$
$E = 127$

IEEE double precision

$\beta = 2$
$t = 52$
$e = -1022$
$E = 1023$

Its not easy to think directly in binary, so lets "translate" the IEEE specifications into base 10 equivalents

IEEE single precision ($\beta = 2$, binary) 4 bytes = 32 bits

$t = 23$ → about 7 or 8 decimal digits

$e = -126$, $E = 127$ → range of numbers is about 10^{-38} to 10^{38}

IEEE double precision ($\beta = 2$, binary) 8 bytes = 64 bits

$t = 52$ → about 15 to 16 decimal digits

$e = -1022$, $E = 1023$ → range of numbers is about 10^{-308} to 10^{308}

Integer: 4 bytes, range = -2^{31} to $2^{31}-1$ { $-2,147,483,648$ to $2,147,483,647$

Conclusions

1) Computers have finite precision. This has non-obvious consequences

2) Storage error size

$0 \leq$ Storage error $\leq \beta^{-t}$ chopping

$-\frac{1}{2}\beta^{-t} \leq$ Storage error $\leq \frac{1}{2}\beta^{-t}$ rounding

3) Storage error and the error in simple binary calculations ($+, -, *, /$) is 1 (binary) digit in the last place

4) Catastrophic cancellation is bad and can be difficult to detect and anticipate.