2. Accuracy and errors

Quantifying error

If x is a computed solution and x^* is the true value,

- Absolute error: $E_a = |x - x^*|$
- Fractional (or relative) error: $E_f = |x - x^*|/|x^*|$

Estimating error

In real problems, we generally know the computed solution, but not the true value. Then we have to estimate the (absolute or fractional) error. One way to do that is by comparing solutions computed using different methods, or the same method but, for example, different step sizes or numbers of iterations.

Practice problem

Find the absolute and relative errors of approximating π by $22/7$.

Practice problem

Find the absolute and relative errors of approximating the derivative of $x^{1.8}$ at $x = 0.5$ using centered finite difference with $h = 0.1$.

Roundoff error

A computation system can only represent some numbers exactly. Numbers input or arising from computations that can't be represented exactly are typically rounded to the closest number that can.

Floating-point numbers

In, for example, *double precision*, which is the most frequently used for numerical computations, each representable (binary) number is expressed in 64 bits as

$$(-1)^s \times 2^{E-1023} \times (1.b_2 b_3 b_4 \ldots b_{53})_2$$

where s is the *sign bit* that determines if a number is positive or negative, the *exponent* E is an 11-bit integer between 0 to 2046 that sets the number's scale, and the 52 bits $b_2 \ldots b_{53}$ form the *mantissa* that specifies the value.

There are other possible formats, such as *single precision*, which is less accurate and can only handle smaller numbers, but takes half as many bits to store.

Machine epsilon

Machine epsilon, ϵ, measures the fractional error due to roundoff in a given computation system. ϵ is defined as the smallest representable number larger than 1, minus 1. For double precision, $\epsilon = 2^{-52} \approx 10^{-16}$

Other characteristics of double-precision floating point

- Only fractions whose denominator is a power of 2 – i.e. which can be written $\frac{I}{2^p}$ for integer I and p – can be represented exactly in binary floating-point
- Special values: $0, -0, \infty$ (`numpy.inf`), $-\infty$, `numpy.nan` (not a number)
- Largest representable number is 2^{1024} (10^{308}) – anything larger *overflows* to `inf`
- Smallest representable positive number is 2^{-1022} (or 2^{-1074} including *denormal* numbers) – anything smaller *underflows* to 0

Practice problem

The expression $f(n) = \frac{n^n}{n!}$, where n is a positive integer, can be calculated in two mathematically equivalent ways:
a) $\frac{(n^n)}{(n!)}$,
b) $(\frac{n}{n})(\frac{n}{n-1})(\frac{n}{n-2})\ldots(\frac{n}{1})$.
Try both methods in Python. For each method, which n leads to overflow?

Propagation of roundoff: multiplication and division

If we have some error at each step of a computation, for example due to roundoff, what happens to the final result?

If we multiply or divide two numbers a, b, one of which has a fractional error ϵ, the fractional error in ab or a/b is still ϵ

Addition and subtraction

On the other hand, if we add or subtract two numbers with some fractional error ϵ, the fractional error in $a + b$ or $a - b$ could in some cases be much larger

That's the case when we subtract two numbers that are very close together (subtractive cancellation)

This is an example of an *ill-conditioned* mathematical procedure – where a small fractional change in one of the inputs can cause a large fractional change in the result

Some roundoff error examples

- `0.3 / 0.1`
- Plotting $(x-1)^7$, expanded, near $x = 1$
- Partial sums of the Taylor series of $\sin(x)$ about $a = 0$ for $x = 41\pi/2$
- Solving linear systems with almost singular coefficient matrices

Practice problem

a) Rewrite $\sqrt{x+1}-1$ to get rid of subtractive cancellation when x is close to 0.
b) Rewrite $\sqrt{x+1}-\sqrt{x}$ to get rid of subtractive cancellation when x is very large.

Practice problem

Use the Taylor series with remainder term proportional to x^6 to approximate the expression $\frac{e^x-1}{x}$ in a form not subject to subtractive cancellation for x close to 0.

Practice problem

a) Use the Taylor series with remainder term proportional to x^5 to approximate the expression $1 - \sin(x)$ in a form not subject to subtractive cancellation for x close to $\pi/2$ radians.
b) Use algebra and trigonometric identities to find an exact mathematical equivalent to $1 - \sin(x)$ that is not subject to subtractive cancellation for x close to $\pi/2$ radians.

Truncation error

- Generally, error in solving a mathematical problem using approximate numerical methods
- More specifically, using only some number of terms from a series such as the Taylor series to derive or apply a numerical method, when theoretically using an infinite number of terms would give the exact answer
- Examples:
  - Newton method (exact in convergent cases after infinite iterations)
  - Finite difference formula (exact if $h = 0$)
  - Euler method (exact if $h = 0$)

Practice problem

Using Python, estimate e^{-7} using the Taylor series about a $= 0$ in two ways: a) Add the first 50 terms in the Taylor series of e^x for $x = -7$. b) Add the first 50 terms in the Taylor series of e^x for $x = 7$ and take the reciprocal. What are the fractional errors of each estimate? Which of the two estimates is more accurate, and why?

Total error

In solving a mathematical problem, both truncation and roundoff errors may occur

Example: Error in the finite difference formula and in the Taylor series sum can be dominated by either truncation or roundoff, depending on h or on the number of terms summed

Additionally, there may be modeling, measurement, or other errors in terms of the mathematical problem representing the engineering situation