Old definition of Numerical analysis: "the study of rounding errors."

This is boring, and not very meaningful.

Better definition - Trefethen '92: "the study of algorithms for the problems of continuous mathematics."

Much of the field of numerical analysis came out of trying to efficiently, and stably, solve $Ax = b$ in floating-point arithmetic.

NA touches all fields now: ODEs, PDEs, physics, etc.

General overview of topics to be covered:
- solving nonlinear systems of equations
- numerical linear algebra
- polynomial interpolation
- numerical integration
- ODEs: initial value problems
- Monte Carlo methods
- Fast Fourier Transform

Then will be computing! Familiarize yourself with MATLAB.

Textbook: Suli & Meyers, Intro to Numerical Analysis (free from NYU)
Many numerical analysis/Math failures can be found at  ima.umn.edu/~arnold/disasters
This is an important field!

**First topic:** Solving a nonlinear equation

**Linear:** \( 3x + 7 = 2 \)  
Can solve by hand, explicit form of solution

**Nonlinear:** \( \cos x + x^2 - 7 = 5 \)  
No closed form solution, must use a numerical method

General form of the problem:

\[
\text{Solve } f(x) = 0 \implies \text{Root finding}
\]

![Graph](image)

Many solutions

![Graph](image)

No solution (at least if \( x \) is required to be real-valued)

I.e., \( x^2 + 1 = 0 \implies x = \pm i \)

A sufficient condition for a solution to exist on the interval \([a,b]\):

- \( f(a) < 0 \) & \( f(b) > 0 \)
- OR \( f(a) > 0 \) & \( f(b) < 0 \)
**Thm:** If \( f \) is continuous and real-valued on \([a, b]\), and if \( f(a) \cdot f(b) < 0 \), then there exists an \( x \in (a, b) \) s.t. \( f(x) = 0 \).

**Proof:** Merely apply the Intermediate Value Thm. (Calc I).

Can we use this Thm to design a numerical method for solving \( f(x) = 0 \)?

**Bisection**

Let \( a_0 = a \), \( b_0 = b \), the original interval.

\([a_1, b_1] \) be the interval obtained after \( l \) splittings.

Then \( b_1 - a_1 = \frac{b_0 - a_0}{2^l} = \frac{L}{2^l} \) with \( L = b_0 - a_0 \).

Let \( x_1 = \frac{a_1 + b_1}{2} \) be our approximation of the solution to \( f(x) = 0 \) on step \( l \).
When do we stop the splittings? How many steps of bisection do we take?

If we want to guarantee that \(|x_n - x^*| < \varepsilon\),

\[
\frac{b_n - a_n}{2^n} \leq \frac{b_0 - a_0}{2^L} \leq \frac{1}{2^{L+1}} \leq \varepsilon
\]

then we need to choose \(L\) such that

\[
| x_n - x^* | \leq \frac{b_n - a_n}{2} = \frac{1}{2} \frac{b_0 - a_0}{2^L} = \frac{1}{2^{L+1}} \leq \varepsilon
\]

\[\Rightarrow 2^{L+1} > \frac{L}{\varepsilon} \Rightarrow L > 1 + \log_2 \frac{L}{\varepsilon} .\]

If \(e_n = \) error on \(n^{th}\) step

\[e_n = | x_n - x^* | = \text{absolute error in } x_n .\]

Then \(e_{n+1} = \frac{1}{2} e_n .\)

\[\Rightarrow \text{The error goes down by a factor of } 2 .\]

This is not very fast.

Bisection only used the sign of the function \(f\) at \(a\) and \(b\).
Can we derive a better (faster) method by using the actual values \(f(a)\) and \(f(b)\)?

Next time: Secant method & Newton’s Method.