Math 260: Python programming in math

Errors and algorithms

Error and residuals

Measuring error

There are two fundamental types of error. Let

$$x = (\text{solution to } f(x) = 0), \quad \tilde{x} = \text{approximation to } x$$

- **Absolute error** $|x - \tilde{x}|$
- **Relative error** $|x - \tilde{x}|/|x|$
  - $|x|$ is very large/small, relative may be a better measure
  - rel. error $\approx 10^{-(k+1)}$ means $\approx k$ significant digits

Lastly. there is an indirect measure...

- The **residual** is $f(\tilde{x})$
  - i.e. the leftover when the approx. is plugged into the equation
- The residual is unreliable, but is **computable**

Measuring error: Example

$$f(x) = (x - 100)^7, \quad x = 100, \qquad \tilde{x} = 100.1$$

- Absolute/relative errors: 0.1 and $0.1/100 = 10^{-3}$

  (Error is in **fourth** digit: $\tilde{x} = 1.001 \times 10^2$; three digits of accuracy).
- Residual:

$$f(\tilde{x}) = (0.1)^7 = 10^{-7} \text{ (deceptively small!)}.$$

When do we use each measure?

- Residual is easy to check - that is its main feature

  ... because it **isn't the same as error**
- Relative error makes sense most of the time (but absolute is easier)
- Example: red light and orange light wavelengths $\lambda = 700$ nm and 600nm; difference 10^{-7}m - 'small' absolute error in these units...

Key theme (for later) - what is error?:

Algorithms take in 'tolerances' that say 'compute to wqithin (this error)'.
The algorithm must answer: **what does 'error' mean**?

Measuring error: sensitivity

In numerics, we care about how **sensitive** a problem is to changes in the input.

For example, consider the problem:

$$\text{given } x_0, \quad \text{evaluate } f(x_0).$$

Suppose the input x_0 is peturbed by a small amount δ to a new value $\tilde{x}_0$....

$$f(\tilde{x}_0) = f(x_0 + \delta) \approx f(x_0) + f'(x_0)\delta$$

The difference in the results is related to δ by

$$|f(\tilde{x}_0) - f(x_0)| \approx |f'(x_0)|\delta.$$

That is, as the error in the **input** propagates to the **result**,
it is **amplified by a factor** $f'(x_0)$:

$$(\text{error in result}) \approx |f'(x_0)|(\text{error in input}).$$

Measuring error: sensitivity

For example, take $f(x) = \tan(x)$ and

$$x_0 = \frac{\pi}{2} - 10^{-2}, \quad \tilde{x}_0 = x_0 - 10^{-2}.$$

Then

$$\tan(x_0) = 99.996\cdots \quad \tan(\tilde{x}_0) = 49.99$$

so the difference 10^{-2} in $x \implies$ difference ≈ 50 in f!

Definition (Condition)

A very sensitive problem in this sense is called **ill-conditioned**.
A not-so sensitive problem is called **well-conditioned**.

The typical 'amplification' factor for the *relative* error is called the **condition number**, defined in a suitable way (depends on problem).

For instance, the condition number for evaluating $f(x)$ is $f'(x_0)/f(x_0)$ since

$$\frac{|f(x_0) - f(\tilde{x}_0)|}{|f(x_0)|} \approx \frac{|f'(x_0)|}{f(x_0)} |x_0 - \tilde{x}_0|.$$

Ill-conditioned problems are inherently hard to solve!
Errors the result are hard to control, even if the inputs are fine.

An algorithm: bisection

Bisection: the setup

The problem (zero-finding):

Let $f(x)$ be a continuous function and suppose there is a solution x^* to

$$f(x) = 0.$$

Goal: Find a **numerical solution** $\tilde{x}$ such that the error is less than a given **tolerance** `tol`, i.e.

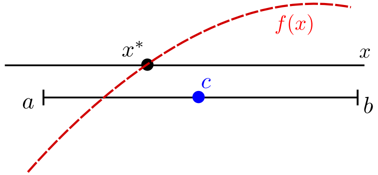
$$|\tilde{x} - x^*| < \texttt{tol}$$

(this is the **absolute error**; the relative error is $|\tilde{x} - x^*|/|x^*|$)

- Requirement: there is a **bracketing interval** $[a, b]$ such that

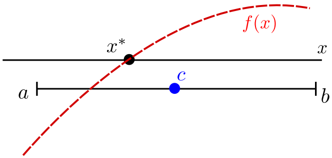
$$f(a) \text{ and } f(b) \text{ have opposite signs.}$$

- By the intermediate value theorem, there is a zero in $[a, b]$



- What guess c minimizes the possible error? $\implies$ take the midpoint

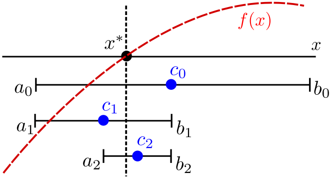
Bisection: the algorithm



- Assume $f(c) \neq 0$. Since $f(a), f(b)$ have opposite signs,

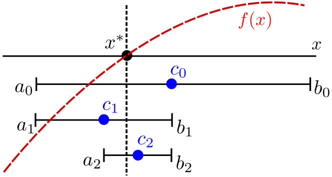
  exactly one of $[a, c]$ and $[c, b]$ is a bracketing interval.

- Now apply the same process to the new bracketing interval.



- Key question: How fast do the midpoints converge to x^*?

Bisection: the algorithm



- Let $[a_n, b_n]$ and $c_n = (a_n + b_n)/2$ denote the interval/midpoint at n-th step
- Bound on the error:

$$|x^* - c_n| < \frac{1}{2}|b_n - a_n|$$

- We know how the interval sizes change, so

$$|x^* - c_n| < \frac{1}{2}2^{-n}|b_0 - a_0|$$

  (intervals halve in width at each step).
- $\implies$ error decreases by a factor of 2 each step (not bad, not great...)
- Benefit: not much is assumed of f