CS450: Numerical Analysis

Howard Hao Yang

Assistant Professor, ZJU-UIUC Institute

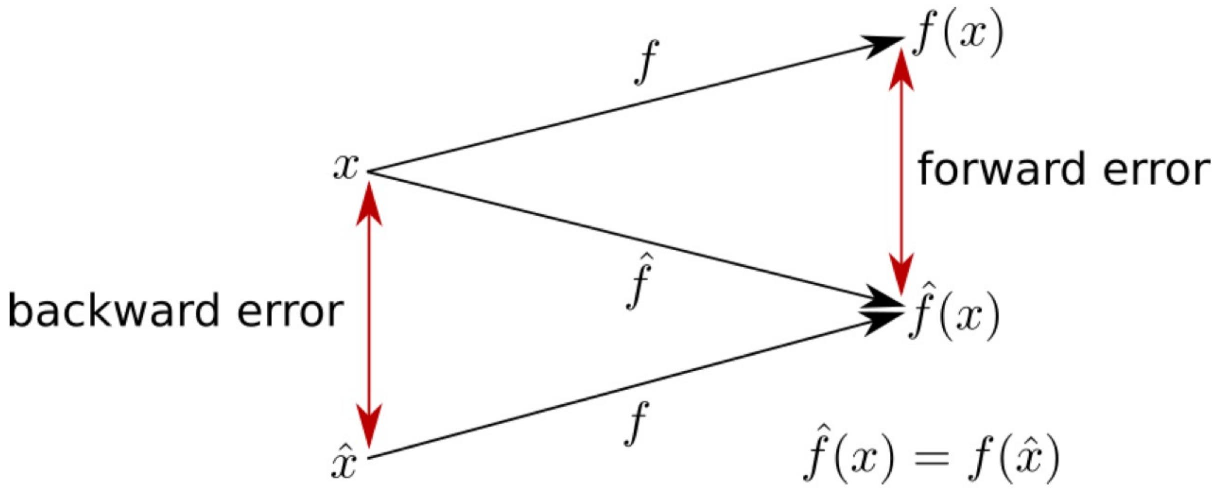
22/09/2023

Approximation Errors

- Last lecture: we want to evaluate $\sin(\frac{\pi}{8})$, but the requirement is without using a computer/calculator
  - We have to opt for approximations
  - Using $\sin\left(\frac{\pi}{8}\right) \approx \frac{\pi}{8} \approx \frac{3.14}{8} = 0.3927$, we have an approximated solution
  - Two types of errors:
    - Truncation error, caused by approximating $\sin(x)$ by x
    - Rounding error, caused by approximating π by 3.14
  - There could be other errors, making things complicated… we want something more unified…

Approximation and Errors

- Forward and backward error
  - Suppose we want to compute $y = f(x)$, where $f\colon R \to R$, but obtain approximate value $\hat{y}$
  - Forward error: $\Delta y = \hat{y} - y$
  - Backward error: $\Delta x = \hat{x} - x$, where $f(\hat{x}) = \hat{y}$



Approximation and Errors

- Forward and backward error
  - Suppose we want to compute $\sin(\frac{\pi}{8})$, and the system produces $\sin\left(\frac{\pi}{8}\right) \approx \frac{\pi}{8} \approx \frac{3.14}{8} = 0.3927$
  - What is the forward error?: $\Delta y = \hat{y} - y =$
  - What is the backward error?: $\Delta x = \hat{x} - x =$
    - Is backward error unique?
    - Why we need backward error?

Approximation and Errors

- Forward and backward error
  - What is forward error?
    - The computational error of an algorithm
    - Essentially, this is what we really want for an algorithm, but usually hard to obtain…
  - What is backward error?
    - Backward error enables us to measure computational error with respect to data propagation error

Conditioning

- Absolute condition number
  - The absolute condition number is a property of the problem, measuring its sensitivity to perturbations in input
  - How much a small change in the input leads to changes in the output
  - Formally, defined by the ratio of absolute errors at output and input

$$\kappa_{abs}(f) = \lim_{\delta \to 0} \max_{\|\Delta x\| < \delta} \frac{\|\Delta f\|}{\|\Delta x\|}$$

Conditioning

- Exercise
  - Absolute condition number, defined by the ratio of absolute errors at output and input

$$\kappa_{abs}(f) = \lim_{\delta \to 0} \max_{\|\Delta x\| < \delta} \frac{\|\Delta f\|}{\|\Delta x\|}$$

  - Calculate the absolute condition number of
    - $f(x) = |x|$ at $x = 0$
    - $f(x) = e^x$ at $x = 1$
  - What do we observe?

Conditioning

- Exercise
  - Absolute condition number, defined by the ratio of absolute errors at output and input

$$\kappa_{abs}(f) = \lim_{\delta \to 0} \max_{\|\Delta x\| < \delta} \frac{\|\Delta f\|}{\|\Delta x\|}$$

  - If f is differentiable at x, the absolute condition number is essentially the derivative at that point

$$\kappa_{abs}(f) = \lim_{\delta \to 0} \max_{\|\Delta x\| < \delta} \frac{\|\Delta f\|}{\|\Delta x\|} = \frac{df}{dx}$$

Conditioning

- Relative condition number
  - Formally, defined by the ratio of relative errors at output and input

$$\kappa_{rel}(f) = \lim_{\delta\to 0} \max_{\|\Delta x\|<\delta} \frac{\|\Delta f / f(x)\|}{\|\Delta x / x\|} = \lim_{\delta\to 0} \max_{\|\Delta x\|<\delta} \frac{\left\| \frac{f(x+\Delta x) - f(x)}{f(x)} \right\|}{\left\| \frac{\Delta x}{x} \right\|}$$

$$= \lim_{\delta\to 0} \max_{\|\Delta x\|<\delta} \frac{\|f(x+\Delta x) - f(x)\|}{\|\Delta x\|} \frac{\|x\|}{\|f(x)\|} = \frac{|x \cdot f'(x)|}{|f(x)|} = \frac{|x| \cdot \kappa_{abs}(f)}{|f(x)|}$$

Conditioning

- Exercise
  - Relative condition number, defined by the ratio of relative errors at output and input

$$\kappa_{rel}(f) = \frac{|x \cdot f'(x)|}{|f(x)|}$$

  - Calculate the absolute condition number of
    - $f(x) = \sqrt{x}$ at $x = 2$
    - $f(x) = e^x$ at $x = 1$

Conditioning

- Interpretation
  - (Relative) condition number

$$\kappa(f) = \frac{|x \cdot f'(x)|}{|f(x)|}$$

  - Condition number indicates how much the system will be amplifying the errors in the input: suppose we are to evaluate $f(x) = \tan(x)$ for x near $\frac{\pi}{2}$
  - Suppose $x_1 = \frac{\pi}{2} - 0.001$ and $x_2 = \frac{\pi}{2} - 0.002$, $|x_1 - x_2| = 0.001$ but $|f(x_1) - f(x_2)| = 500$
  - What is the condition number?

Conditioning

- Interpretation
  - (Relative) condition number

$$\kappa(f) = \frac{|x \cdot f'(x)|}{|f(x)|}$$

  - Linking the forward and backward error

$$\text{forward\_error} = \kappa(f) \times \text{backward\_error}$$

  - Condition number indicates how much the system will be amplifying the errors in the input
  - Forward error may not be easy to obtain, but condition number and backward errors are accessible

Conditioning

- Example
  - Evaluate the value of $\sqrt{2}$
  - Algorithm: make guess on t such that $t^2 \leq 2$ until reaching some desire precision
  - Equivalently, it means evaluating $f(x) = \sqrt{x}$ at $x = 2$
  - To start with, we approximate $\sqrt{2}$ by $t = 1.4$
    - Forward error: $\sqrt{2} - 1.4 =?$
    - Backward error: $2 - (1.4)^2 =?$
    - Condition number: $\kappa(f) = \frac{|x \cdot f'(x)|}{|f(x)|} =?$

Conditioning

- Example
  - Evaluate the value of $\sqrt{2}$
  - Algorithm: make guess on t such that $t^2 \le 2$ until reaching some desire precision
  - Equivalently, it means evaluating $f(x) = \sqrt{x}$ at $x = 2$
  - Next, we approximate $\sqrt{2}$ by $t = 1.41$
    - Forward error: $\sqrt{2} - 1.41 = ?$
    - Backward error: $2 - (1.41)^2 = ?$
    - Condition number: $\kappa(f) = \frac{|x \cdot f'(x)|}{|f(x)|} = ?$

Conditioning

- Remark
  - (Relative) condition number

$$\kappa(f) = \frac{|x \cdot f'(x)|}{|f(x)|}$$

  - Depends on the function f itself, not on anything else
  - By applying an algorithm, we evaluate $f(x)$ by $\hat{f}(x) = \hat{y}$
  - Using backward error analysis, we assume $\hat{f}(x) = f(\hat{x})$ for some $\hat{x}$ that is close to x, i.e., our approximated solution to the original problem is the exact solution to a "nearby" point

Conditioning

- Remark
  - Condition number provides a sense about errors due to round-off, which might result in useless/unreliable results
  - If condition number is 1, then 1% input error → 1% output error
  - If condition number is 1000, then 0.1% input error → 100 amplification in output
- Conditioning tells only half the story…
  - Even if a problem is well conditioned, does it mean that we are bound to evaluate it well?

Algorithm Stability

- Example
  - To evaluate $a_n = 2^{-n}$, use the following recursion
    - $a_0 = 1$, $a_1 = 1/2$
    - $a_{n+1} = \frac{5}{2} a_n - a_{n-1}$
  - This algorithm works on the paper
  - What if there is a very small perturbation in a_0? Say $a_0 = 1 + 2\times 10^{-16}$

Algorithm Stability

- Example
  - To evaluate $a_n = 2^{-n}$, use the following recursion
    - $a_0 = 1 + 2\times10^{-16}$, $a_1 = 1/2$
    - $a_{n+1} = \frac{5}{2}a_n - a_{n-1}$
  - A piece of code in Matlab

```
delta = 2e-16;

A(1) = 1+delta;
A(2) = 1/2;

N = 100;

for i = 2:N
    A(i+1) = 5/2*A(i) - A(i-1);
end
```

51	52	53	54	55	56	57	58	59	60	61	62	63	64
-0.0833	-0.1667	-0.3333	-0.6667	-1.3333	-2.6667	-5.3333	-10.6667	-21.3333	-42.6667	-85.3333	-170.6667	-341.3333	-682.6667

Algorithm Stability

- Characterization
  - Algorithm is stable if result produced is relatively insensitive to perturbations during computation
  - Stability of algorithms is analogous to conditioning of problems: Small changes in the initial data produce correspondingly small changes in the final results
  - Accuracy: closedness of computed solution to true solution of the problem
  - Applying stable algorithm to well-conditioned problem yields accurate solution

Algorithm Stability

- Characterizing stability
  - Suppose that $E_0 > 0$ denotes an error introduced at some stage in the calculation and E_n represents the magnitude of the error after n subsequent operations
    - If $E_n \approx CnE_0$, where C is a constant independent of n, then the growth of error is linear;
    - If $E_n \approx C^n E_0$, for some $C > 1$, then the growth of error is called exponential
  - Linear growth of error is usually unavoidable, and when E_0 and C are small, the results are generally acceptable
  - Exponential growth of error should be avoided because the terms C^n becomes large for even relatively small values of n

Algorithm Stability

- Example
  - To evaluate $a_n = 2^{-n}$, use the following recursion
    - $a_0 = 1$, $a_1 = 1/2$
    - $a_{n+1} = \frac{5}{2} a_n - a_{n-1}$
  - What would be the growth of error?
  - Step 1: check that for any constants c_1 and c_2, $a_n = c_1 \cdot \left(\frac{1}{2}\right)^n + c_2 \cdot 2^n$ is a solution to the above recursion

Algorithm Stability

- Example
  - To evaluate $a_n = 2^{-n}$, use the following recursion
    - $a_0 = 1,\ a_1 = 1/2$
    - $a_{n+1} = \frac{5}{2}a_n - a_{n-1}$
  - Step 2: check that for any constants c_1 and c_2, $a_n = c_1 \cdot \left(\frac{1}{2}\right)^n + c_2 \cdot 2^n$ is a solution to the above recursion; and if $a_0 = 1,\ a_1 = 1/2$, then $c_1 = 1$ and $c_2 = 0$

Algorithm Stability

- Example
  - To evaluate $a_n = 2^{-n}$, use the following recursion
    - $a_0 = 1,\ a_1 = 1/2$
    - $a_{n+1} = \frac{5}{2} a_n - a_{n-1}$
  - Step 3: check that for any constants c_1 and c_2, $a_n = c_1 \cdot \left(\frac{1}{2}\right)^n + c_2 \cdot 2^n$ is a solution to the above recursion; and if $a_0 = 1 + \delta$, $a_1 = 1/2$, then $c_1 =?$ and $c_2 =?$

Algorithm Stability

- Example
  - To evaluate $a_n = 2^{-n}$, use the following recursion
    - $a_0 = 1,\ a_1 = 1/2$
    - $a_{n+1} = \frac{5}{2} a_n - a_{n-1}$
  - Step 4: if $a_0 = 1 + \delta,\ a_1 = 1/2$, then $c_1 =$? and $c_2 =$?; what is the growth of error?

Learning Objectives

- Forward error and backward error
- Conditioning number of a problem
- Algorithm stability