Introduction to Numerical Analysis: Conditioning and Stability

Mohammad Sadegh Talebi

Department of Computer Science
University of Copenhagen

Conditioning

Conditioning informally captures how sensitive the solution of a problem may be to small relative changes in the input data.

Conditioning

A problem is said to be well-conditioned if small changes in the problem parameters lead to small changes in the solution. It is called badly conditioned or ill-conditioned if small changes in the parameters can cause large changes in the solution.

- The solution of a badly conditioned problem is very sensitive to changes in the parameters.
- Importance: in practical situations, data are almost always subject to uncertainty, so it is important to have an idea of problem condition.

Conditioning

The previous *informal* definition can be made more precise for certain problems by defining a condition number.

Informally speaking, a condition number can be generically defined as

$$\text{condition number at } x = \frac{\text{relative error in output corresponding to } x}{\text{relative error in } x}$$

And

$$\text{condition number} = \sup_{\text{all } x} \frac{\text{relative error in output corresponding to } x}{\text{relative error in } x}$$

- Typically, in both definitions supremum is taken over all small enough perturbations of x
- The lager the condition number, the more ill-conditioned the problem.

Conditioning: Function Evaluation

Problem: To evaluate f at x.

If x is slightly perturbed (by Δx), what is the effect on $f(x)$?

For small enough Δx: $$\frac{f(x+\Delta x)-f(x)}{f(x)} \approx \frac{\Delta x \cdot f'(x)}{f(x)}$$

Now,

$$\text{condition number at } x = \frac{\text{relative error in } f(x)}{\text{relative error in } x}$$

$$= \sup_{\text{small } \Delta x} \left| \frac{\big(f(x+\Delta x)-f(x)\big)/f(x)}{\Delta x/x} \right| \approx \left| \frac{x f'(x)}{f(x)} \right|$$

Conditioning: Function Evaluation

Example: Evaluating $f(x) = \ln(x)$

$$\text{condition number at } x \approx \left| \frac{x f'(x)}{f(x)} \right|$$

$$\text{condition number at } x = \frac{1}{|\ln(x)|}$$

$\ln(1) = 0$, so near $x = 1$, small relative errors in x may lead to large relative errors in $\ln(x)$.

Conditioning: Linear Equations

Problem: To solve $Ax = b$ with A non-singluar. The solution is $x = A^{-1}b$.

How sensitive x is to small perturbations in b?

Conditioning: Linear Equations

$$\text{condition number at } b = \frac{\text{relative error in } x}{\text{relative error in } b} = \frac{\|\Delta x\|/\|x\|}{\|\Delta b\|/\|b\|}$$

(x and b are vectors, so we look at norms of them.)

$$A(x + \Delta x) = b + \Delta b \Longrightarrow \Delta x = A^{-1}\Delta b$$

$$\text{condition number} = \sup_{\|b\| \neq 0,\ \text{small } \|\Delta b\| \neq 0} \frac{\|A^{-1}\Delta b\|/\|A^{-1}b\|}{\|\Delta b\|/\|b\|}$$

$$= \sup_{\Delta b \neq 0} \frac{\|A^{-1}\Delta b\|}{\|\Delta b\|} \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

$$= \|A\| \cdot \|A^{-1}\|$$

This leads to define the condition number $\kappa(A) = \|A\| \cdot \|A^{-1}\|$, where $\|\cdot\|$ is a matrix norm.

Conditioning: Root Finding

Problem: Finding the root of a polynomial $f \in C^2$ (i.e., f'' is continuous everywhere).

If we perturb f to $f + \varepsilon g$, how is a simple root r of f perturbed?

We have $f(r + \Delta r) + \varepsilon g(r + \Delta r) = 0$. Assuming that $g \in C^2$, Taylor's Approximation gives:

$$\Big(\underbrace{f(r)}_{=0} + f'(r)\Delta r + \tfrac{1}{2}\underbrace{(\Delta r)^2}_{\ll \Delta r} f''(\xi)\Big) + \varepsilon\Big(g(r) + g'(r)\Delta r + \tfrac{1}{2}\underbrace{(\Delta r)^2}_{\ll \Delta r} g''(\xi)\Big) = 0$$

$$\Delta r \approx -\frac{\varepsilon g(r)}{f'(r) + \varepsilon g'(r)} \approx -\varepsilon \frac{g(r)}{f'(r)}$$

Conditioning: Root Finding

Example (Wilkinson's Polynomial) Consider $f(x) = \prod_{k=1}^{20}(x-k)$, which has simple roots $r = 1, 2, \ldots, 20$. How is the root $r = 20$ affected by perturbing f to $f + \varepsilon x^{20}$?

$$\Delta r \approx -\varepsilon \frac{g(20)}{f'(20)} = -\varepsilon \frac{20^{20}}{19!} \approx -10^9 \varepsilon$$

We can use the previous slide to define the corresponding condition number.

Stability

Numerical algorithms almost never compute the exact solution due to rounding error, introduced by the finite precision used by computers

So it is crucial to have an idea whether an algorithm magnifies and propagates such rounding errors or not.

Numerical Stability

A numerical process/algorithm is numerically stable if the inevitable small errors introduced during the calculations lead to small errors in the result. It is called unstable if small errors during the calculation can lead to very large errors in the result.

Stability vs. Conditioning

Conditioning:

- A property of a problem
- Not always under control
- In ill-conditioned problems with large errors in parameters error, the solution will be inaccurate, regardless of the method.

Stability:

- A property of an algorithm
- Might be under control
- An unstable algorithm produces inaccurate result by introducing *unnecessarily* large errors.

In 1950's few people could distinguish between condition and stability.

Subtracting Nearly Equal Numbers

Many unstable behaviours occur when subtracting nearly equal numbers.

In many cases, errors can be avoided by using alternative form of functions.

Example 1: How to compute $y = \sqrt{x^2+3} - x$ for large x?

For large x, $\sqrt{x^2+3} \approx x$.

$$y = \left(\sqrt{x^2+3} - x\right) \frac{\sqrt{x^2+3}+x}{\sqrt{x^2+3}+x} = \frac{3}{\sqrt{x^2+3}+x}$$

Roots of Quadratic Equations

Example 2: Solve $ax^2 + bx + c = 0$ with $a \neq 0$.

Algorithm 1: Compute

$$x^+ = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x^- = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Algorithm 1 is unstable when $b^2 \gg 4|ac|$:

- If $b > 0$, potential of lagre error for x^+
- If $b < 0$, potential of lagre error for x^-

Solve $x^2 + (10^8 + 10^{-8})x + 1 = 0$ (Roots: $x = -10^8, -10^{-8}$).
Python using Algorithm 1 gives: $-1.4901161193847656 \times 10^{-8}$ and -100000000.0

Roots of Quadratic Equations

Algorithm 2:

- **Case 1: $b > 0$.** x^- is accurately computed and

$$x^+ = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \times \frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}}$$
$$= \frac{4ac}{2a\big(-b - \sqrt{b^2 - 4ac}\big)} = \frac{c}{ax^-}$$

- **Case 2: $b < 0$.** x^+ is accurately computed and

$$x^- = \frac{c}{ax^+}$$

Now Python using Algorithm 2 gives accurate solutions: -10^{-8} and -100000000.0

Area of a Triangle

Example 3: Compute the area of a triangle with sides of length $a, b,$ and c.

Algorithm 1 (Horn's Formula):

$$A = \sqrt{p(p-a)(p-b)(p-c)}, \quad p = \tfrac{1}{2}(a+b+c)$$

a, b, c	$10, 8, 3$	$1, 1, 10^{-13}$	$1, \cos(2^{-52}), \sin(2^{-52})$
A_{exact}	9.921567	5.000000×10^{-14}	1.110223×10^{-16}
A_{Heron}	9.921567	4.996004×10^{-14}	0.000000

In needle-shaped triangles where a is close to $b + c$, Horn's formula yields inaccurate result (why?).

Area of a Triangle

Example 3: Compute the area of a triangle with sides of length a, b, and c.

Algorithm 2 (Kahan's Formula (1983)): W.l.o.g. assume $a \geq b \geq c$. Then

$$A = \tfrac{1}{4}\sqrt{\big(a+(b+c)\big)\big(c-(a-b)\big)\big(c+(a-b)\big)\big(a+(b-c)\big)}$$

a, b, c	$10, 8, 3$	$1, 1, 10^{-13}$	$1, \cos(2^{-52}), \sin(2^{-52})$
A_{exact}	9.921567	5.000000×10^{-14}	1.110223×10^{-16}
A_{Heron}	9.921567	4.996004×10^{-14}	0.000000
A_{Kahan}	9.921567	5.000000×10^{-14}	1.110223×10^{-16}

In contrast to Heron's Formula, Kahan's Formula gives accurate result for all data.

Subtracting Nearly Equal Numbers

Example 4: How to compute $y = 1 - \cos x$ for x close to 0?

Using Taylor's expansion of $\cos x$ around 0:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots$$

So, ignoring terms with powers more than 8 yields

$$y = 1 - \cos x \approx \frac{x^2}{2!} - \frac{x^4}{4!} + \frac{x^6}{6!} - \frac{x^8}{8!}$$

Subtracting Nearly Equal Numbers

An even wiser way:

$$\begin{aligned}\frac{x^2}{2}-\frac{x^4}{4!}+\frac{x^6}{6!}-\frac{x^8}{8!} &= \frac{x^2}{2}\left(1-\frac{x^2}{3\times 4}+\frac{x^4}{3\times 4\times 5\times 6}-\frac{x^6}{3\times 4\times\cdots\times 8}\right)\\ &= \frac{x^2}{2}\left(1-\frac{x^2}{12}\left(1-\frac{x^2}{5\times 6}+\frac{x^4}{5\times 6\times 7\times 8}\right)\right)\\ &= \frac{x^2}{2}\left(1-\frac{x^2}{12}\left(1-\frac{x^2}{30}\left(1-\frac{x^2}{7\times 8}\right)\right)\right)\end{aligned}$$

Equivalently,

$$t_1 = \frac{x^2}{2}$$

$$t_{n+1} = -\frac{x^2}{(2n+1)(2n+2)}t_n \quad n\geq 1$$

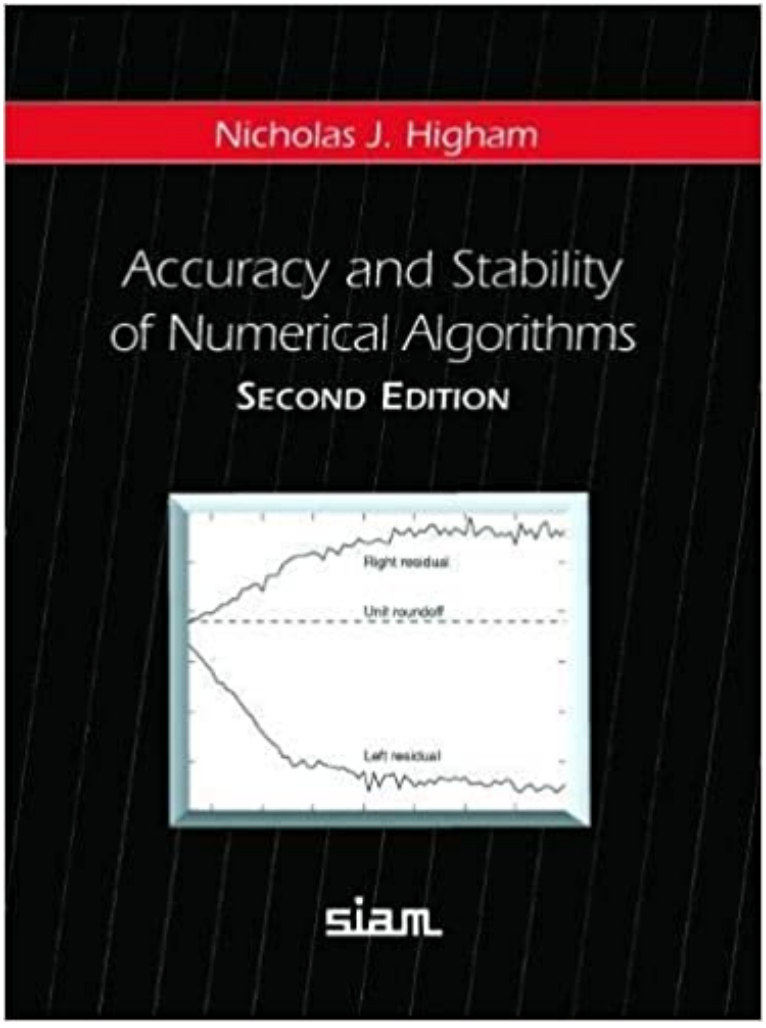
Then, $s_n = \sum_{k=1}^{n} t_k$ gives an approximation to $1-\cos x$.

Numerical Stability

Numerical stability is an active area of research within numerical analysis.

An interesting book on this topic:

Accuracy and Stability of Numerical Algorithms by Nicholas Higham



References

- David Kincaid and Ward Cheney, *Numerical Analysis: Mathematics of Scientific Computing*, American Mathematical Society, 2009.
- Lieven Vandenberghe, *Lecture Notes for Applied Numerical Computing (ECE133A)*, UCLA, 2011.
- Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 2002.