# MATH3511/6111: Scientific Computing

## 09. Conditioning and Stability

Lindon Roberts

Semester 1, 2022

Office: Hanna Neumann Building #145, Room 4.87
Email: lindon.roberts@anu.edu.au
Based on lecture notes written by S. Roberts, L. Stals, Q. Jin, M. Hegland, K. Duru.

Australian National University

# Conditioning and Stability

Throughout this course, we are looking at finding approximate solutions to mathematical problems.

In practice, we will then implement algorithms on a computer, which introdues rounding errors.

- Conditioning is about how sensitive mathematical problems are to perturbations.
- Stability is about how sensitive algorithms are to perturbations/rounding errors

# Abstract Framework

All the mathematical processes we discuss can be viewed as a mapping from data to solution. We can formalise this:

- $X$ is a vector space where our inputs (data) live.
- $Y$ is a vector space where our solution lives.
- $\boldsymbol{f} : X \to Y$ returns the true mathematical solution for given input data.

We are interested in how $\boldsymbol{f}$ behaves near our actual input data (call it $\boldsymbol{x}$).

# Conditioning

**Definition (Well-conditioned)**

A problem is well-conditioned at $\boldsymbol{x}$ if any small perturbations of the data $\boldsymbol{x}$ cause only small changes in the solution $\boldsymbol{f}(\boldsymbol{x})$.

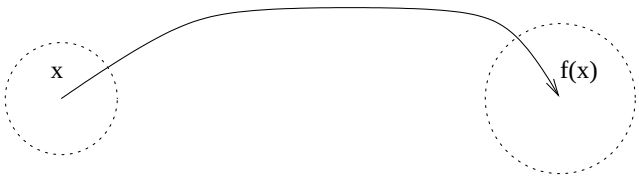


**Figure 1.** Well-conditioned problem: small perturbations in the data get mapped to small perturbations in the solution.

# Conditioning

**Definition (Ill-conditioned)**

A problem is ill-conditioned at $\boldsymbol{x}$ if some small perturbations of the data $\boldsymbol{x}$ cause large changes in the solution $\boldsymbol{f}(\boldsymbol{x})$.

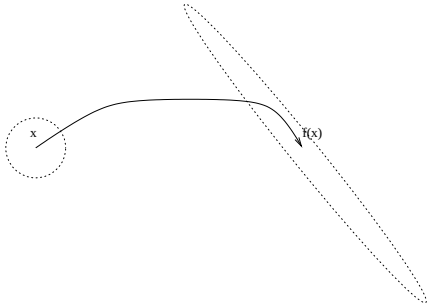


**Figure 2.** Ill-conditioned problem: some small perturbations in the data get mapped to large perturbations in the solution.

# Condition Number

**Question**

How can we quantify the concept of well/ill-conditioned problems?

Suppose our original data $\boldsymbol{x}$ is perturbed to $\boldsymbol{x} + \delta\boldsymbol{x}$. This gives us two (nearby) problems with solutions $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{f}(\boldsymbol{x} + \delta\boldsymbol{x})$. The change in solution is

$$\delta\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x} + \delta\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}).$$

**Definition (Absolute condition number)**

The absolute condition number for the problem $\boldsymbol{f}$ at a point $\boldsymbol{x}$ is

$$\widetilde{\kappa} = \max_{\delta\boldsymbol{x}} \frac{\|\delta\boldsymbol{f}\|}{\|\delta\boldsymbol{x}\|},$$

where the maximum is taken over all (infintesmially small) $\delta\boldsymbol{x}$.

# Condition Number

Recall from Taylor's theorem that if $\boldsymbol{f}$ is differentiable,

$$\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\delta}) = \boldsymbol{f}(\boldsymbol{x}) + J(\boldsymbol{x})\boldsymbol{\delta x} + \mathcal{O}(\|\boldsymbol{\delta x}\|^2),$$

where $J(\boldsymbol{x})$ is the Jacobian matrix of $\boldsymbol{f}$ at $\boldsymbol{x}$.

Then since $\boldsymbol{\delta f} \approx J(\boldsymbol{x})\boldsymbol{\delta x}$, we have

$$\widetilde{\kappa} = \|J(\boldsymbol{x})\|.$$

# Condition Number

However, like all our other calculations, we always try to normalise our changes (e.g. relative errors). The relative condition number measure the relative changes in the solution vs. relative changes in the inputs.

**Definition (Relative condition number)**

The relative condition number for the problem $\boldsymbol{f}$ at a point $\boldsymbol{x}$ is

$$\kappa = \max_{\boldsymbol{\delta x}} \frac{\|\boldsymbol{\delta f}\|/\|\boldsymbol{f}(\boldsymbol{x})\|}{\|\boldsymbol{\delta x}\|/\|\boldsymbol{x}\|},$$

where the maximum is taken over all (infintesmially small) $\boldsymbol{\delta x}$.

If $\boldsymbol{f}$ is differentiable, then

$$\kappa = \frac{\|J(\boldsymbol{x})\|}{\|\boldsymbol{f}(\boldsymbol{x})\|/\|\boldsymbol{x}\|}.$$

# Condition Number

**Warning**

The condition number only depends on the actual mathematical problem we care about. What algorithm we choose to find the solution is irrelevant!

It is a fundamental part of the problem: no algorithm can get around ill-conditioning (without changing the mathematical problem it solves).

In general, small $\kappa$ means well-conditioned and large $\kappa$ means ill-conditioned, but there is no clear cutoff between them.

# Condition Number: Examples

**Problem**

Given a positive number, multiply it by 3.

Here, our input is $x > 0$ and our solution is $f(x) = 3x$.

Our solution map is differentiable, $J(x) = f'(x) = 3$. Therefore the (relative) condition number is

$$\kappa = \frac{\|J(x)\|}{\|f(x)\|/\|x\|} = \frac{3}{|3x|/|x|} = 1.$$

Since $\kappa$ is not large, the problem is well-conditioned (for all $x$).

# Condition Number: Examples

**Problem**

Given a positive number, calculate its square root.

Here, our input is $x > 0$ and our solution is $f(x) = \sqrt{x}$.

Our solution map is differentiable, $J(x) = f'(x) = 1/(2\sqrt{x})$. Therefore,

$$\kappa = \frac{\|J(x)\|}{\|f(x)\|/\|x\|} = \frac{|1/(2\sqrt{x})|}{|\sqrt{x}|/|x|} = \frac{1}{2}.$$

Since $\kappa$ is not large, the problem is well-conditioned (for all $x$).

Here, the absolute condition number $\widetilde{\kappa} = \|J(x)\|$ can get very large for $x \approx 0$. However, we are always well-conditioned in terms of relative changes.

# Condition Number: Examples

**Problem**

Given two numbers, calculate their difference.

Our input is $\boldsymbol{x} \in \mathbb{R}^2$ and our solution is $f(\boldsymbol{x}) = x_1 - x_2$. For convenience, let's use the $\infty$-norm on $\mathbb{R}^2$.

The Jacobian matrix is $J(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & -1 \end{bmatrix}$. The operator $\infty$-norm of this matrix is the maximum absolute row sum: $\|J(\boldsymbol{x})\|_\infty = |1| + |-1| = 2$.

Then our condition number is

$$\kappa = \frac{\|J(x)\|_\infty}{\|f(x)\|_\infty / \|x\|_\infty} = \frac{2}{|x_1 - x_2| / \max(|x_1|, |x_2|)} = \frac{2\max(|x_1|, |x_2|)}{|x_1 - x_2|}.$$

If $|x_1 - x_2|$ is small compared to $|x_1|$ or $|x_2|$, then the problem is ill-conditioned.

This is exactly when catastrophic cancellation occurs! (see floating point lectures)

# Wilkinson's Polynomial

A famous early example of ill-conditioning is for finding the roots of polynomials.

**Problem**

Given $p(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + x^n$, find the roots of $p(x)$.

Wilkinson's polynomial (James Wilkinson, 1963) is the simple-looking polynomial

$$p(x) = (x-1)(x-2)\cdots(x-20).$$

Clearly $p(x)$ has 20 roots at $x = 1, 2, \ldots, 20$.

Wilkinson wanted to test his implementation of Newton's method for rootfinding, so he ran it on the monomial form of $p(x)$:

$$p(x) = x^{20} - 210x^{19} + \cdots$$

Despite his code being correct, he could not find the roots!

# Wilkinson's Polynomial

Suppose we want to find the 15th root ($x_{15} = 15$) as a function of the coefficient $a_{15}$.

With some work, it can be shown that the condition number is huge:

$$\kappa = \frac{a_{15} 15^{14}}{5!\, 14!} \approx 5.1 \times 10^{13}.$$

Suppose we are working in double precision, where $\epsilon \approx 2.22 \times 10^{-16}$ Assuming the relative error in the stored value of $a_{15}$ is $\epsilon$, then the relative error in the root is

$$\frac{|\delta x_{15}|}{|x_{15}|} \approx \kappa \frac{|\delta a_{15}|}{|a_{15}|} \leq \kappa\epsilon \approx 1.13 \times 10^{-2}.$$

That is, 16 correct digits in $a_{15}$ leads to only 2 correct digits in $x_{15}$, regardless of how good our rootfinding algorithm is!

(Wilkinson was only working in single precision, where $\epsilon \approx 1.1 \times 10^{-7}$, so he got no digits of accuracy in his solution)

# Condition Number: Examples

An important example of conditioning is matrix-vector multiplication.

**Problem**

Given $\boldsymbol{x} \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$, compute $A\boldsymbol{x}$.

Here, the solution is $\boldsymbol{y} = A\boldsymbol{x}$, and the perturbed solution is

$$\boldsymbol{y} + \delta\boldsymbol{y} = A(\boldsymbol{x} + \delta\boldsymbol{x}) = A\boldsymbol{x} + A\delta\boldsymbol{x},$$

and so $\delta\boldsymbol{y} = A\delta\boldsymbol{x}$.

The condition number for this problem is

$$\kappa = \max_{\delta\boldsymbol{x}} \frac{\|\delta\boldsymbol{y}\|/\|\boldsymbol{y}\|}{\|\delta\boldsymbol{x}\|/\|\boldsymbol{x}\|} = \max_{\delta\boldsymbol{x}} \frac{\|A\delta\boldsymbol{x}\|/\|A\boldsymbol{x}\|}{\|\delta\boldsymbol{x}\|/\|\boldsymbol{x}\|}$$

# Condition Number: Examples

Noting that $\|A\boldsymbol{\delta x}\| \leq \|A\| \cdot \|\boldsymbol{\delta x}\|$, we get

$$\kappa \leq \frac{\|A\|\|\boldsymbol{x}\|}{\|A\boldsymbol{x}\|}.$$

If $A$ is invertible, then $\frac{\|\boldsymbol{x}\|}{\|A\boldsymbol{x}\|} = \frac{\|A^{-1}(A\boldsymbol{x})\|}{\|A\boldsymbol{x}\|} \leq \|A^{-1}\|$, and so

$$\kappa \leq \|A\| \cdot \|A^{-1}\|.$$

**Definition (Condition number of a matrix)**

The condition number of an invertible matrix is

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|.$$

This number appears in the analysis of many linear algebra problems.

# Stability

Conditioning tells us how sensitive our mathematical problem is to perturbations in the input space.

In practice, we cannot evaluate the true solution $\boldsymbol{f}(\boldsymbol{x})$, but instead run some algorithm (implemented on a computer) which approximately finds the solution, $\widetilde{\boldsymbol{f}}(\boldsymbol{x})$.

We hope that our algorithm is good, for example as measured by absolute or relative errors

$$\|\widetilde{\boldsymbol{f}}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x})\| \qquad \text{or} \qquad \frac{\|\widetilde{\boldsymbol{f}}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x})\|}{\|\boldsymbol{f}(\boldsymbol{x})\|}.$$

No matter the problem, there will always be some error in $\widetilde{\boldsymbol{f}}(\boldsymbol{x})$, even if it is just from storing the data $\boldsymbol{x}$ in finite precision.

If the problem is ill-conditioned (like Wilkinson's polynomial), this can be enough to cause large errors in $\widetilde{\boldsymbol{f}}(\boldsymbol{x})$ regardless of the algorithm! So, we need some notion of a "good algorithm" that is achievable.

# Stability

**Definition (Stable algorithm)**

An algorithm $\widetilde{\boldsymbol{f}}$ is stable if, for all data $\boldsymbol{x} \in X$,

$$\frac{\|\widetilde{\boldsymbol{f}}(\boldsymbol{x}) - \boldsymbol{f}(\widetilde{\boldsymbol{x}})\|}{\|\boldsymbol{f}(\widetilde{\boldsymbol{x}})\|} = \mathcal{O}(\epsilon),$$

for some $\widetilde{\boldsymbol{x}}$ with

$$\frac{\|\widetilde{\boldsymbol{x}} - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \mathcal{O}(\epsilon).$$

"A stable algorithm gives nearly the right answer to nearly the right question." (Nick Trefethen & David Bau)

In practice, we can often do a bit better than this.

# Stability

**Definition (Backward stable algorithm)**

An algorithm $\widetilde{\boldsymbol{f}}$ is backward stable if, for all data $\boldsymbol{x} \in X$,

$$\widetilde{\boldsymbol{f}}(\boldsymbol{x}) = \boldsymbol{f}(\widetilde{\boldsymbol{x}}),$$

for some $\widetilde{\boldsymbol{x}}$ with

$$\frac{\|\widetilde{\boldsymbol{x}} - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \mathcal{O}(\epsilon).$$

"A backwards stable algorithm gives exactly the right answer to nearly the right question." (Nick Trefethen & David Bau)

# Stability: Examples

All standard operations $+$, $-$, $\times$, $\div$ are backwards stable. Let's consider subtraction.

**Algorithm**

Subtract two numbers using the algorithm

$$\widetilde{f}(x_1, x_2) = \text{fl}(\text{fl}(x_1) - \text{fl}(x_2)).$$

$$\begin{aligned}\widetilde{f}(x_1, x_2) &= \text{fl}(x_1(1+\delta_1) - x_2(1+\delta_2)),\\ &= [x_1(1+\delta_1) - x_2(1+\delta_2)](1+\delta_3),\\ &= x_1 \underbrace{(1+\delta_1)(1+\delta_3)}_{=1+\delta_4} - x_2 \underbrace{(1+\delta_2)(1+\delta_3)}_{=1+\delta_5}.\end{aligned}$$

So $\widetilde{f}(x_1, x_2) = \widetilde{x}_1 - \widetilde{x}_2 = f(\widetilde{x}_1, \widetilde{x}_2)$ where $\widetilde{x}_1 = x_1(1+\delta_4)$ and $\widetilde{x}_2 = x_2(1+\delta_5)$.

Since $|\delta_4|, |\delta_5| \leq 2\epsilon + \mathcal{O}(\epsilon^2)$, subtraction is backwards stable.

# Stability: Examples

**Algorithm**

Calculate the eigenvalues of a matrix by

1. Calculate the characteristic polynomial and expand it into the form $p(x) = a_0 + a_1 x + \cdots + a_n x^n$.
2. Use a rootfinding algorithm to calculate the roots of $p(x)$.

The first step will introduce (small) rounding errors in the coefficients $a_i$ of $p(x)$.

We know from Wilkinson's polynomial that small errors in the $a_i$ can introduce large errors in the roots (since the problem is ill-conditioned).

Therefore the overall algorithm is not stable.

# Stability: Examples

For example, suppose we want to find the eigenvalues of

$$A = \begin{bmatrix} 1 + 10^{-14} & 0 \\ 0 & 1 \end{bmatrix}.$$

The characteristic polynomial is

$$p(x) = \det(xI - A) = x^2 + (-2 - 10^{-14})x + (1 + 10^{-14}).$$

Using NumPy's polynomial rootfinding function `numpy.roots`, I get

$$x = 0.9999999850988439 \qquad \text{and} \qquad x = 1.0000000149011663$$

Despite the coefficients of $p(x)$ being exactly correct to machine precision (16 digits), I only get about 8 digits of accuracy in the eigenvalues (compare: `numpy.linalg.eig` gives 16 digits accuracy).

# Stability

Why is backwards stability useful?

Well-conditioned problem + backwards stable algorithm = small relative error

To be more formal:

**Theorem**

*Suppose a backwards stable algorithm $\widetilde{\boldsymbol{f}}$ is used to solve a problem $\boldsymbol{f}$ with input data $\boldsymbol{x}$ and condition number $\kappa(\boldsymbol{x})$. Then for machine precision $\epsilon$,*

$$\frac{\|\widetilde{\boldsymbol{f}}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x})\|}{\|\boldsymbol{f}(\boldsymbol{x})\|} = \mathcal{O}(\kappa(\boldsymbol{x})\epsilon).$$