Conditioning Theory, Backward Stability, and Error Analysis

This document covers conditioning theory, backward stability, and the theoretical foundations of error analysis.

1 Types of Errors in Computation

1. **Measurement errors** – in the initial data
2. **Modeling errors** – approximating reality with mathematics
3. **Method errors** – discretization, truncation in algorithms
4. **Rounding errors** – finite precision arithmetic

Key principle: Numerical algorithms should be designed with rounding errors in mind from the beginning, not as an afterthought.

2 Well-Posed vs Ill-Posed Problems

Definition 2.1 (Well-Posed Problem)**.** A problem (P): find x such that $F(x) = y$ for given y is **well-posed** if:

1. A solution $x = F^{-1}(y)$ exists
2. The solution is unique
3. The solution depends continuously on the data y (equivalent to F^{-1} being continuous)

If any condition fails, the problem is **ill-posed** (e.g., solving $Ax = b$ with singular A).

3 Error Measurement

3.1 For Scalars

Given an approximation $\hat{x}$ of a nonzero real number x:

- **Absolute error**: $E_a(\hat{x}) = |x - \hat{x}| = |\Delta x|$
- **Relative error**: $E_r(\hat{x}) = \frac{|x-\hat{x}|}{|x|} = \frac{|\Delta x|}{|x|}$

3.2 For Vectors

Given $\hat{x} \in \mathbb{R}^n$ approximating $x \in \mathbb{R}^n$:

- **Absolute error**: $E_a(\hat{x}) = |\Delta x|$
- **Relative error** (two variants):
  - **Normwise**: $|\Delta x|_g = \frac{|\Delta x|}{|x|}$
  - **Componentwise**: $|\Delta x|_c = \max_i \frac{|\Delta x_i|}{|x_i|}$ (when $x_i \neq 0$)

4 Conditioning Theory

4.1 General Condition Number

For a well-posed problem where $x = G(y)$ with $G = F^{-1}$:

If there's an error Δy in the data, the computed value is $\hat{x} = G(y + \Delta y)$.

For sufficiently small Δy:

$$\hat{x} - x \approx G'(y) \cdot \Delta y \tag{1}$$

This leads to the **relative error relationship**:

$$\frac{|\hat{x} - x|}{|x|} = K(G, y) \cdot \frac{|\Delta y|}{|y|} + O(|\Delta y|^2) \tag{2}$$

where the **condition number** is:

$$K(G, y) = \left| \frac{yG'(y)}{G(y)} \right| \tag{3}$$

Interpretation: The condition number measures how much relative errors in the input are amplified in the output.

4.2 Condition Number for Polynomial Evaluation

4.2.1 Case 1: Perturbing the evaluation point z

For $p(z) = \sum_{i=0}^{n} a_i z^i$ with $z \neq 0$:

$$K(p, z) = \frac{|zp'(z)|}{|p(z)|} \tag{4}$$

Key observation: The condition number approaches infinity as z approaches a root of p. Thus, evaluating a polynomial near its roots is inherently ill-conditioned.

4.2.2 Case 2: Perturbing the coefficients

When coefficients $a = (a_0, \ldots, a_n)^T$ are perturbed by Δa with:

- Data norm: $|\Delta a|_D = \max_{i=0:n} |\Delta a_i|/|a_i|$ (relative componentwise)
- Result norm: $|\Delta x|_R = |\Delta x|/|x|$ (relative absolute)

The **condition number** is:

$$K(p(z), a) = \frac{\sum_{i=0}^{n} |a_i z^i|}{|p(z)|} = \frac{\tilde{p}(|z|)}{|p(z)|} \tag{5}$$

where $\tilde{p}(|z|) = \sum_{i=0}^{n} |a_i||z|^i$ is the polynomial with absolute value coefficients.
Important: This condition number is also particularly large near roots of p.

4.3 Conditioning for Linear Systems

For the system $Ax = b$ where A is nonsingular:
If x solves $Ax = b$ and $\hat{x} = x + \Delta x$ solves $Ax = b + \Delta b$:

$$A\Delta x = \Delta b \implies \Delta x = A^{-1}\Delta b \tag{6}$$

This gives:

$$|\Delta x| \leq |A^{-1}||\Delta b| \tag{7}$$

Since $|Ax| = |b| \geq \frac{|b|}{|A|}$, we have $|x| \geq \frac{|b|}{|A|}$
Therefore:

$$\frac{|\Delta x|}{|x|} \leq |A||A^{-1}|\frac{|\Delta b|}{|b|} \tag{8}$$

4.4 Matrix Condition Number

The **condition number** of matrix A is:

$$\kappa(A) = |A||A^{-1}| \tag{9}$$

4.4.1 Perturbations in A

If $x + \Delta x$ solves $(A + \Delta A)x = b$:

$$\frac{|\Delta x|}{|x|} \leq \kappa(A)\frac{|\Delta A|}{|A|} \tag{10}$$

4.4.2 Perturbations in both A and b

If $x + \Delta x$ solves $(A + \Delta A)x = b + \Delta b$:

$$\frac{|\Delta x|}{|x|} \leq \frac{\kappa(A)}{1 - \kappa(A)\frac{|\Delta A|}{|A|}} \left(\frac{|\Delta A|}{|A|} + \frac{|\Delta b|}{|b|} \right) \tag{11}$$

4.5 Remarks on Condition Numbers

1. **Norm dependence**: The condition number depends on the choice of norm
2. **Distance to singularity**: Generally measures the inverse distance to singularity
3. **Problem property**: Depends only on the problem, not on the algorithm
4. **First-order analysis**: Only considers infinitesimal perturbations

5 Forward vs Backward Error Analysis

5.1 Forward Error Analysis

Approach: Track the propagation of rounding errors through each operation in algorithm $\hat{G}$ applied to input y.
Result: Provides an upper bound on the gap between exact solution x and computed solution $\hat{x}$ (the forward error).
Answers: "To what accuracy is the problem solved?"
Disadvantage: Tracking intermediate error propagation becomes complicated quickly and leads to expressions that are difficult to exploit.

5.2 Backward Error Analysis

Approach: A two-stage process:
Stage 1: Identify the computed approximation $\hat{x}$ as the exact evaluation of G at perturbed data $(y + \Delta y)$:

$$\hat{x} = G(y + \Delta y) \tag{12}$$

The error Δy is the **backward error**. This answers: "Which problem was actually solved?"
Stage 2: Since the backward error Δy is estimated or bounded, analyze the effect using conditioning theory:

$$\text{forward error} \lesssim \text{condition number} \times \text{backward error} \tag{13}$$

Advantages:

- Cleaner analysis than forward error
- Separates algorithm reliability (backward error) from problem difficulty (conditioning)
- Provides natural definition of stability

5.3 The Backward Error

The **backward error** associated with computed solution $\hat{x} = \hat{G}(y)$ is:

$$\eta(\hat{x}) = \min_{\Delta y \in D} \{|\Delta y|_D : \hat{x} = G(y + \Delta y)\} \tag{14}$$

It measures the smallest perturbation to the data that makes $\hat{x}$ the exact solution.

5.4 Key Relationship

At first order:

$$\text{forward error} \lesssim \text{cond}(P, y) \times \eta(\hat{x}) \tag{15}$$

6 Stability of Algorithms

6.1 Definition: Backward Stability

An algorithm is **backward stable** for solving problem (P) if the computed solution $\hat{x}$ has a small backward error $\eta(\hat{x})$.

More specifically, an algorithm is backward stable in finite precision (with unit roundoff u) if:

$$\eta(\hat{x}) = O(u) \tag{16}$$

6.2 Interpretation

A backward-stable algorithm:

- Computes the **exact solution** of a slightly perturbed problem
- The perturbation is small enough that the perturbed and exact problems are indistinguishable at the working precision
- Introduces no more error than is intrinsic to representing the data in finite precision
- Makes optimal use of the available computer precision

Important: Backward stability does NOT guarantee that the solution is accurate—only that it's as accurate as the conditioning of the problem allows.

6.3 Accuracy of Backward-Stable Algorithms

For a backward-stable algorithm with backward error $\eta(\hat{x}) \approx u$:

$$|\Delta x| \lesssim K \cdot u \tag{17}$$

where K is the condition number.

Ill-conditioned problem: A problem with relative accuracy u is ill-conditioned if its condition number K satisfies:

$$K \times u \geq 1 \tag{18}$$

In this case, even a backward-stable algorithm may produce results with large forward error.

7 Standard Model for Floating-Point Arithmetic

7.1 Rounding Function

Let $\mathrm{fl} : \mathbb{R} \to \mathbb{F}$ map real numbers to floating-point numbers.

A **rounding** satisfies:

1. $\mathrm{fl}(x) = x$ for all $x \in \mathbb{F}$ (exactness)
2. $\mathrm{fl}(x) \leq \mathrm{fl}(y)$ for all $x \leq y$ (monotonicity)

Rounding modes:

- **Round to nearest**: $\mathrm{fl}(x) = \arg\min_{y \in \mathbb{F}} |x - y|$
- **Directed rounding**: toward 0, $+\infty$, or $-\infty$

7.2 Fundamental Theorem

Theorem 7.1. For any $x \in \mathbb{R}$:

$$\mathrm{fl}(x) = x(1 + \delta), \quad |\delta| \leq u \tag{19}$$

where u is the **unit roundoff**:

- $u = \varepsilon/2$ for round-to-nearest
- $u = \varepsilon$ for directed rounding

Here $\varepsilon = \beta^{1-p}$ is the machine epsilon (p = precision, β = base).

7.3 Standard Model for Operations

For basic operations $\circ \in \{+, -, \times, /\}$:
Multiplicative form:

$$\mathrm{fl}(x \circ y) = (x \circ y)(1 + \delta), \quad |\delta| \le u \tag{20}$$

Divisive form:

$$\mathrm{fl}(x \circ y) = \frac{x \circ y}{1 + \delta'}, \quad |\delta'| \le u \tag{21}$$

These are equivalent to first order since $(1 + \delta)^{-1} \approx 1 - \delta$ for small δ.

7.4 Model With Underflow

To account for possible underflow (or overflow):

$$\mathrm{fl}(x \circ y) = (x \circ y)(1 + \delta) + \eta \tag{22}$$

where $|\delta| \le u$, $|\eta| \le \underline{u}$, and $\delta \cdot \eta = 0$.

- If underflow occurs: $\delta = 0$, $\eta \ne 0$
- Otherwise: $\eta = 0$, $\delta \ne 0$

For IEEE 754 double precision (binary64) with round-to-nearest:

- $u = 2^{-53} \approx 1.11 \times 10^{-16}$
- $\underline{u} = 2^{-1074} \approx 4.94 \times 10^{-324}$

8 Summary of Key Concepts

8.1 The Three Pillars of Error Analysis

1. **Condition Number**: Measures problem difficulty (sensitivity to input perturbations)
2. **Backward Error**: Measures algorithm reliability (equivalent input perturbation)
3. **Forward Error**: Measures solution accuracy (actual error in result)

8.2 The Fundamental Relationship

$$\text{Forward Error} \lesssim \text{Condition Number} \times \text{Backward Error} \tag{23}$$

This separates:

- **What we can't control**: Condition number (inherent to the problem)
- **What we can control**: Backward error (depends on algorithm design)

8.3 Design Principles

1. **For well-conditioned problems** ($K \approx 1$): Most reasonable algorithms work well
2. **For ill-conditioned problems** ($K \gg 1$):
   - Backward stability is essential
   - Even then, accuracy may be limited by $K \cdot u$
   - Reformulating the problem may help
3. **Ultimate goal**: Backward-stable algorithms that achieve $\eta(\hat{x}) = O(u)$

8.4 Practical Implications

- **Backward stability** is a gold standard for algorithm quality
- A backward-stable algorithm is "as good as it gets" given the problem's conditioning
- Poor accuracy with a backward-stable algorithm indicates an ill-conditioned problem, not a bad algorithm
- Sometimes problem reformulation is necessary to avoid ill-conditioning

This framework provides a systematic way to:

- Analyze numerical algorithms
- Understand limitations of finite precision
- Design robust numerical software
- Diagnose sources of inaccuracy in computations