

CS450: Numerical Analysis¹

Howard Hao Yang

Assistant Professor, ZJU-UIUC Institute



¹These slides have been modified from Michael T. Heath's material for the book "Scientific Computing: An Introductory Survey"

Lecture 1

Introduction²

² "Drink a cup of coffee. Test whose lecture you can follow during 30 mn. This defines a cut in the set of mathematicians. This cut measures the strength of the coffee." – *Coffee Meter*, constructed by Adrien Douady (1935—2006)



Scientific Computing

A little bit information about the instruction team

- Instructor: Howard H. Yang
 - ▶ Joined ZJUI in Dec. 2020
 - ▶ Research interests: (stochastic) models for the understanding and design of Internet of Things networks and distributed machine learning systems
 - ▶ I like things that *scales*, namely, they shall be tiny, low-cost, and easy-to-be-used, hence easily to spread
- The teaching assistants
 - ▶ Chenhao Wang, master student at ZJUI
 - ▶ Zeshen Li, PhD student at ZJUI

Scientific Computing

How about you?

- Why did you take this course?
- What do you want to learn from it?

Motivating Examples

Computation is the resource that has most transformed our age...



- Video game rendering.

Motivating Examples

Computation is the resource that has most transformed our age...

Default (GPT-3.5)

RO

I am going to introduce ChatGPT in my numerical analysis course, can give me a motivation to do that?



Introducing ChatGPT into your numerical analysis course can be a highly motivating and beneficial decision for both you as an instructor and your students. Here are some compelling reasons to consider:



1. Real-world Relevance: Numerical analysis is a fundamental discipline in computational mathematics with numerous real-world applications. Incorporating ChatGPT allows you to connect your course to cutting-edge technologies and their practical use. Many industries, including finance, engineering, and data science, rely on numerical analysis, and AI-driven tools like ChatGPT are increasingly part of the toolkit.
2. Hands-On Experience: ChatGPT can serve as a hands-on tool for students to apply numerical analysis concepts in practical scenarios. You can design exercises and projects that involve using ChatGPT to solve numerical problems or analyze data, providing students with a tangible application of their learning.

- Large Language models: extremely expensive in training.

Motivating Examples

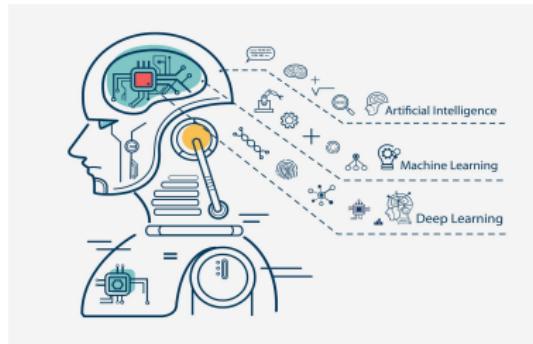
Computation is the resource that has most transformed our age...



- Computing is everything (?)

Motivating Examples

Computation is the resource that has most transformed our age...



- Machine learning models
 - ▶ Much of the computation consumption is due to problems like solving linear equations, evaluating integrals/derivatives, or finding the minimum of (highly) nonlinear functions.
 - ▶ What is common in these problems is that their solutions have no analytic form. Methods to compute such numbers, by a computer or on paper, are of an approximate nature.

Outline

1) Scientific Computing

2) Approximation and Error

3) Floating-Point Arithmetic

Scientific Computing

Introduction

- What is this course all about?
 - ▶ 'Scientific Computing' (a.k.a. 'Numerical Analysis') describes a family of approaches to obtain approximate solutions to problems *once they've been stated mathematically*.
- Relevant applications
 - ▶ Engineering simulations: weather forecasting, drag from flow over airplane wings, behavior of photonic devices,...
 - ▶ Machine learning: train statistical models based on a collection of data samples with unknown parameters
 - ▶ Image/audio processing: enlargement or filtering

Scientific Computing

What problems do we study, and how?

- Problems with real number, i.e., *continuous* problems
 - ▶ As opposed to *discrete* problems.
 - ▶ Including: How can we put a real number into a computer? (and with what restrictions?)
- What's the general approach?
 - ▶ Pick a general representation (e.g., a polynomial).
 - ▶ Existence/uniqueness?
 - ▶ Develop an algorithm to solve it.

Scientific Computing

Two important questions.

- *How good* of an answer can we expect to our problem?
 - ▶ Can't even represent numbers exactly.
 - ▶ Answers will always be approximate.
 - ▶ It is natural to ask *how far off the mark* we really are.
- *How fast* can we expect the computation to complete?
 - ▶ What algorithms do we use? (what is an algorithm, BTW?)
 - ▶ What is the cost of these algorithms?
 - ▶ Are they efficient (i.e., do they make good use of available machine time)?
- Many times, numerical analysis is about the trade-off between accuracy and efficiency.

Scientific Computing

Course Structure

- Complex problems are reduced to simpler problems
 - ▶ continuous → discretized
 - ▶ nonlinear → linear
 - ▶ complicated problems → polynomials
 - ▶ differential equations → linear equations
- Correspondingly, the contents we shall cover are
 - ▶ error analysis
 - ▶ matrix computing
 - ▶ numerical optimization
 - ▶ ordinary differential equation (ODE) and partial differential equation (PDE)

Scientific Computing

Grading Policy

- Assessment is based on
 - ▶ Attendance (record & quizzes): 10%
 - ▶ Homework (approximately bi-weekly, posted on BB): 30%
 - ▶ Midterm (open book): 15%
 - ▶ Project (paper reading, presentation, and report): 10%
 - ▶ Final (open book): 35%

Scientific Computing

Prerequisites

- Not much specific courses, but...
 - ▶ A solid foundation on linear algebra
 - ▶ A certain level of (multi-variable) calculus skill
 - ▶ Strong programming skills
 - ▶ Not scared of mathematics: Many derivations in class

Scientific Computing

What problems can we study in the first place?

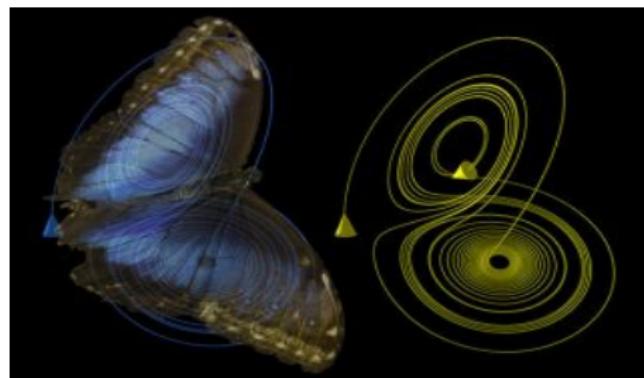
- Problem is *well-posed* if solution
 - ▶ exists
 - ▶ is unique
 - ▶ depends continuously on problem data

Otherwise, problem is *ill-posed*

- These are termed the three *Hadamard criteria*
 - ▶ Proposed by Jacques-Salomon Hadamard in 1923 when he considered the issue of delineating between useful problems and those that weren't worth anything scientifically (at that time).
- Examples of ill-posed problems
 - ▶ Solution to $\frac{df(t)}{dt} = \frac{3}{2}(f(t))^{1/3}$, $f(0) = 0$
 - ▶ Solution to $\frac{df(t)}{dt} = \frac{2-f(t)}{t}$

Scientific Computing

Caveat about well-posedness.



- Even if the problem is well posed, a solution may still be *sensitive* to input data
 - ▶ Is a chaotic system well-posed or ill-posed?
- Computational algorithm should not make sensitivity worse

Approximation

Example: Approximations

- Computing surface area of Earth using formula $A = 4\pi r^2$ involves several approximations
 - ▶ Earth is modeled as sphere, while its true shape is not
 - ▶ Value for radius is based on empirical measurements and previous computations
 - ▶ Value for π requires truncating infinite process
 - ▶ Values for input data and results of arithmetic operations are rounded in computer

Approximation

Sources of Approximation

- Before computation
 - ▶ modeling (George Box: All models are wrong, but some are useful)
 - ▶ empirical measurements
 - ▶ previous computations
- During computation
 - ▶ truncation or discretization
 - ▶ rounding
- Accuracy of final result reflects all these
- Uncertainty in input may be amplified by problem
- Perturbations during computation may be amplified by algorithm

Approximation

How do we measure error?

- *Absolute error* : approximate value – true value
- *Relative error* : $\frac{\text{absolute error}}{\text{true value}}$
- Equivalently, approx value = (true value) \times (1 + rel error)
- True value usually unknown, so we *estimate* or *bound* error rather than compute it exactly
- Relative error often taken relative to approximate value, rather than (unknown) true value

Approximation

Data Error and Computational Error

- Example: Calculate $\sin(\pi/8)$ without a computer/calculator

Approximation

Data Error and Computational Error

- Typical problem: compute value of function $f : \mathbb{R} \rightarrow \mathbb{R}$ for given argument
 - ▶ x = true value of input
 - ▶ $f(x)$ = desired result
 - ▶ \hat{x} = approximate (inexact) input
 - ▶ \hat{f} = approximate function actually computed
- Total error: $\hat{f}(\hat{x}) - f(x) =$

$$\hat{f}(\hat{x}) - f(\hat{x}) + f(\hat{x}) - f(x)$$

computational error + propagated data error

Approximation

Truncation Error and Rounding Error

- Computational error, i.e., errors made during computations, usually consists of truncation error and rounding error
- *Truncation error* : difference between true result (for actual input) and result produced by given algorithm using exact arithmetic
 - ▶ Due to approximations such as truncating infinite series or terminating iterative sequence before convergence
- *Rounding error* : difference between result produced by given algorithm using exact arithmetic and result produced by same algorithm using limited precision arithmetic
 - ▶ Due to the inexact representation of real numbers and arithmetic operations upon them

Approximation

Example: Finite Difference Approximation

- Error in finite difference approximation

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

exhibits a tradeoff between rounding error and truncation error

- Elementary calculus tells us: let $h \rightarrow 0$, we can have the derivative. Is that true?

Approximation

Example: Finite Difference Approximation (cont'd)

- Error in finite difference approximation

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

- Taylor expansion of $f(x)$ gives

$$f(x+h) = f(x) + f'(x)h + \frac{f''(\theta)h^2}{2}, \quad \theta \in [x, x+h]$$

- Suppose evaluating f incurs an error of ϵ and denote by

$$M = \max_{\theta \in [x, x+h]} |f''(\theta)|$$

- Total error is

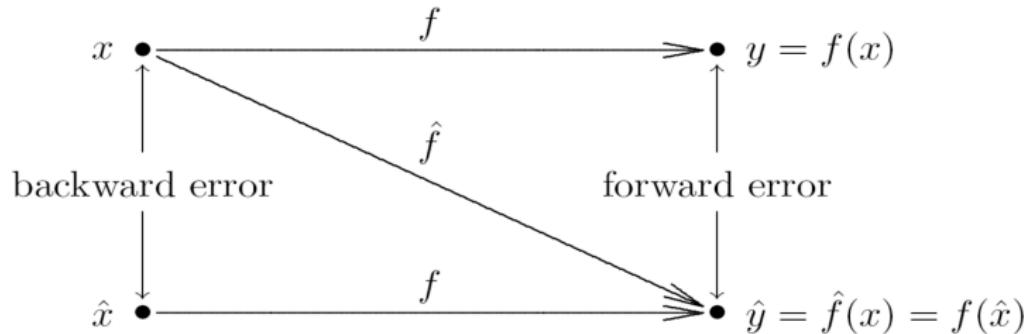
$$\frac{f(x+h) - f(x)}{h} - f'(x) \leq \frac{2\epsilon}{h} + \frac{Mh}{2}$$

- The error is minimized when $h = \sqrt{M\epsilon}$

Approximation

Forward and Backward Error

- Suppose we want to compute $y = f(x)$, where $f : \mathbb{R} \rightarrow \mathbb{R}$, 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

What are forward and backward errors, and why do we need these?

Approximation

Is backward error unique? How do we quantify their relations?

Approximation

Absolute and relative condition number

Approximation

Condition number and error analysis

Approximation

Example

Approximation

How can we estimate algorithm forward error in general?

Approximation

Example: ill-conditioned problem