Lecture 2: Approximations and Errors

BT 2020 – Numerical Methods for Biology

Karthik Raman
Department of Biotechnology, Bhupat & Jyoti Mehta School of Biosciences
Initiative for Biological Systems Engineering (IBSE)
Robert Bosch Centre for Data Science and Artificial Intelligence (RBC-DSAI)

Indian Institute of Technology Madras

https://home.iitm.ac.in/kraman/lab/
https://web.iitm.ac.in/ibse/
https://rbcdsai.iitm.ac.in/

Strategies for Computational Solutions

Problem Reduction

- Infinite-dimensional spaces $\rightsquigarrow$ Finite-dimensional spaces
- Integrals $\rightsquigarrow$ Finite sums
- Derivatives $\rightsquigarrow$ Finite differences
- Differential equations $\rightsquigarrow$ Algebraic Equations
- Non-linear Problems $\rightsquigarrow$ Linear Problems
- Complicated functions $\rightsquigarrow$ Simple functions, e.g. polynomials
- ...

Approximations in Scientific Computation

- Engineering is all about approximation!
- So is scientific computing
  - actually, we have no choice!
  - we need to use discrete computer systems and representations to work on continuous and *infinite* quantities!
- We need to compute accurately, using (limited) *finite precision* arithmetic!

Sources of Approximation

Before Computation

- The model itself!
- Measurement errors
  - e.g. arising out of instrument imprecision
- Previous computations

During Computation

- Truncation / discretisation
- Rounding
  - Finite precision computation

Approximation: An Example

- What is the surface area of the earth?

$$A = 4\pi r^2$$

Can you list the approximations involved?

Approximation: An Example

- What is the surface area of the earth?

$$A = \pi r^2$$

Some approximations

- Shape of the earth!
- How do we measure the radius?
  - Empirically?
  - Based on other computations?
- Value of π!
- Precision of the computers used
- ...

Absolute and Relative Errors

- Obviously, significance of an error is related to magnitude of measured quantity

$$\text{absolute error} = \text{approximate value} - \text{true value}$$

$$\text{relative error} = \frac{\text{absolute error}}{\text{true value}}$$

- Obviously, relative error is undefined if true value is zero
- Relative error can also be expressed in %

Precision vs. Accuracy

- Precision: Number of digits with which a number is expressed (recall significant figures?)
- Accuracy: Number of *correct significant* digits
- If an approximate value has an error of $\approx 10^{-p}$, then its decimal representation has $\approx p$ correct significant digits
- e.g. 3.61803398874989484820458683436563811772030 is a precise number, but not an accurate representation for π
- Often, we may not know the true value itself — if we did, we may not need to approximate it!

Data Error and Computational Error

- Let's consider 1D problems to begin with, e.g. $f : \mathbb{R} \longrightarrow \mathbb{R}$
- Let the true value of the input be $x \Rightarrow$ desired true result is $f(x)$
- But, our input may be inexact, say $\hat{x}$
- Also, our function computation may be approximate, say $\hat{f}$

$$\begin{aligned}
\text{Total error} &= \hat{f}(\hat{x}) - f(x) \\
&= \left(\hat{f}(\hat{x}) - f(\hat{x})\right) + \left(f(\hat{x}) - f(x)\right) \\
&= \text{computational error} + \text{propagated data error}
\end{aligned}$$

Data Error and Computational Error

$$\begin{aligned}
\text{Total error} &= \hat{f}(\hat{x}) - f(x) \\
&= \left(\hat{f}(\hat{x}) - f(\hat{x})\right) + \left(f(\hat{x}) - f(x)\right) \\
&= \text{computational error} + \text{propagated data error}
\end{aligned}$$

- Here, $\hat{f}(\hat{x}) - f(\hat{x})$ is the difference between exact and approximate functions for the *same* input — pure computational error
- $f(\hat{x}) - f(x)$ denotes the difference between exact function values due to error in the input — propagated data error
  - Choice of algorithm has no impact on this!

Data Error and Computational Error

An Example

- Suppose we want a back-of-the-envelope calculation for $\sin(\pi/8)$
- How will you approximate it, without a calculator?
- What is the computational error? What is the data error?

Data Error and Computational Error

An Example

- Suppose we want a back-of-the-envelope calculation for $\sin(\pi/8)$
- How will you approximate it, without a calculator?
- What is the computational error? What is the data error?

- $\sin(\pi/8) \approx \sin(3/8) \approx 3/8 = 0.3750(!)$
- Using a calculator, $\sin(\pi/8) = 0.382683432... \approx 0.3827$
- $\text{Total error} = \hat{f}(\hat{x}) - f(x) = 0.3750 - 0.3827 = -0.0077$
- Propagated data error, arising out of inexact input is $f(\hat{x}) - f(x) = \sin(3/8) - \sin(\pi/8) \approx 0.3663 - 0.3827 = -0.0164$
- Computational error, cause by truncating infinite series is $\hat{f}(\hat{x}) - f(\hat{x}) = 3/8 - sin(3/8) \approx 0.3750 - 0.3663 = 0.0087$
- In this case, the errors are of opposing signs, offsetting one another!
- In other cases, they may reinforce one another!
- How to get more accurate??

Truncation Error vs. Rounding Error

Computational error can be further split into:

Truncation/Discretisation Error

- Difference between the true result (for actual input) and the result that would be produced by a given algorithm using *exact / infinite-precision* arithmetic
- The algorithm may truncate an infinite series (e.g. Taylor series), replace derivatives by finite differences etc.

Rounding Error

- Difference between the result produced by a given algorithm using exact arithmetic and the result produced by the same algorithm using finite-precision rounded arithmetic
- Arises out of the inexactness in representation of real numbers and arithmetic operations on these numbers

Truncation Error

Example — Finite Difference Approximation

For a differentiable function $f : \mathbb{R} \to \mathbb{R}$, consider the finite difference approximation to the first derivative,

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

By Taylor's theorem,

$$f(x+h) = f(x) + hf'(x) + f''(\theta)h^2/2$$

for some $\theta \in [x, x+h]$. So, the truncation error of the finite difference approximation is bounded by $Mh/2$, where M is a bound on $|f''(t)|$ for t near x.

Rounding Error

Example — Finite Difference Approximation

If the error in function values in bounded by ϵ, the rounding error in evaluating the finite difference formula is bounded by $2\epsilon/h$. The total computation errors is therefore:

$$\boxed{\frac{Mh}{2} + \frac{2\epsilon}{h}}$$

Clearly, as we decrease h, one term increases, while the other decreases $\Rightarrow$ there's a trade-off.
The above function has its minimum at $h = 2\sqrt{\epsilon/M}$.

How to Reduce Error?

- *Truncation error* can be reduced by using better approximations, e.g. more terms in the expansion
- For instance, a more accurate finite difference formula can be used:

$$\boxed{f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}}$$

- *Rounding error* can be reduced by working with higher-precision arithmetic (harder?)

Errors in Practice

- Although both truncation and rounding errors are important, in practice, one or the other tends to dominate
- Roughly speaking,
  - Rounding error dominates purely algebraic problems with finite solution algorithms, while
  - Truncation error dominates in problems involving integrals, derivatives, non-linearities etc. that require a theoretically infinite solution process
- The distinctions made among different types of errors are important for understanding the behaviour of numerical algorithms
- However, in practice, it is not necessary (or even possible!) to precisely delineate the different individual errors — advantageous to lump them all together

Evaluating Complex Functions — Taylor Series

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{(2n+1)}}{(2n+1)!}$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{(2n)}}{(2n)!}$$

N^{th}-order Taylor polynomial for $y = f(x)$ at x_0 is:

$$p_N(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \ldots + \frac{f^{(N)}(x_0)}{N!}(x - x_0)^N$$

$$\Rightarrow p_N(x) = \sum_{n=0}^{N} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

Truncation Error — Exercise

Use **Taylor Series** expansions with $n = 0 \ldots 6$ to approximate $f(x) = cos(x)$ at $x_{i+1} = \pi/3$ on the basis of the value of $f(x)$ and its derivatives at $x_i = \pi/4$.

Forward Error and Backward Error

- Garbage in $\rightsquigarrow$ Garbage out!
- If input data are accurate to only four significant digits, we can expect no more than four significant digits in computed result, no matter how accurate an algorithm we use!
- Suppose we want to compute $y = f(x)$ (again $f : \mathbb{R} \to \mathbb{R}$) — we obtain an approximate value $\hat{y}$
- The discrepancy between the computed and true values, $\Delta y = \hat{y} - y$ is called *forward error*
- This is often difficult to compute ...

Forward Error and Backward Error

- Alternately, let's consider the approximate solution obtained to be the exact solution for a *modified* problem
- Now, "How large a modification to the original problem is required to give the result actually obtained?"
- Or, "How much data error in the initial input x would be required to explain all of the error in the output?"
- $\Delta x = \hat{x} - x$, where $f(\hat{x}) = \hat{y}$ is the *backward error*

$$
\begin{tikzcd}
x \arrow[rr, "f"] \arrow[d, "\text{backward error}"'] \arrow[drr, "\hat{f}"] & & y = f(x) \arrow[d, "\text{forward error}"] \\
\hat{x} \arrow[rr, "f"] & & \hat{y} = \hat{f}(x) = f(\hat{x})
\end{tikzcd}
$$

- Note that the equality $f(\hat{x}) = \hat{f}(x)$ is due to the choice of $\hat{x}$ — this requirement defines $\hat{x}$

Forward Error and Backward Error

$$\begin{tikzcd}[column sep=huge, row sep=huge]
x \arrow[r, "f"] \arrow[d, "\text{backward error}"'] \arrow[dr, "\hat{f}"] & y = f(x) \arrow[d, "\text{forward error}"] \\
\hat{x} \arrow[r, "f"] & \hat{y} = \hat{f}(x) = f(\hat{x})
\end{tikzcd}$$

Example

- As an approximation to $y = \sqrt{2}$, let us use $\hat{y} = 1.4$
- $|\Delta y| = |\hat{y} - y| = |1.4 - 1.4142\ldots| \approx 0.0142 (\approx 1\%)$
- What is the $\hat{x}$ that would give the value of 1.4? $1.4^2 = 1.96$
- Backward error $= |1.96 - 2| = 0.04 (2\%)$

Forward Error and Backward Error

Another Example

Consider $y = \cos(x)$. What are the errors in computing $\cos(x)$ using the truncated Taylor expansion $1 - x^2/2$, for $x = \pi/3$

Sensitivity and Conditioning

- Difficulties in solving a problem accurately are not always due to an ill-conceived formula or algorithm, but may be inherent in the problem being solved
- Even with exact computation, the solution to the problem may be highly sensitive to perturbations in the input data
- A problem is said to be *insensitive*, or *well-conditioned*, if a given relative change in the input data causes a reasonably commensurate relative change in the solution
- A problem is said to be *sensitive*, or *ill-conditioned*, if the relative change in the solution can be much larger than that in the input data
- More formally, we define the condition number of a problem f at x as

Sensitivity and Conditioning

Condition Number

$$\text{Condition number} = \frac{|\text{relative change in solution}|}{|\text{relative change in input data}|}$$

$$\text{Cond} = \frac{|(f(\hat{x}) - f(x))/f(x)|}{|(\hat{x} - x)/x|} = \frac{|(\hat{y} - y)/y|}{|(\hat{x} - x)/x|} = \frac{|\Delta y/y|}{|\Delta x/x|}$$

$$|\text{Relative forward error}| = |\text{condition number}| \times |\text{Relative backward error}|$$

Thus, the condition number can be interpreted as an "amplification factor" that relates forward error to backward error. If a problem is *ill-conditioned* (large condition number), then the relative forward error (perturbation in solution) can be large even if the backward error (relative perturbation in input) is small.

Condition Number for a Differentiable Function

$$\text{Absolute forward error} = f(x+\Delta x) - f(x) \approx f'(x)\Delta x$$

$$\text{Relative forward error} = \frac{f(x+\Delta x) - f(x)}{f(x)} \approx \frac{f'(x)\Delta x}{f(x)}$$

$$\text{Condition number} \approx \frac{f'(x)\Delta x / f(x)}{\Delta x / x} = \left| \frac{x f'(x)}{f(x)} \right|$$

Stability and Accuracy

- The concept of *stability* of a computational algorithm is analogous to conditioning of a mathematical problem
- Both deal with the effects of perturbations
- Stability — refers to the effects of *computational error on the result* computed by an algorithm
- Conditioning — refers to the effects of *data error on the solution* to a problem
- An algorithm is stable if the result it produces is relatively insensitive to perturbations due to approximations made during the computation

Stability and Accuracy

- From the viewpoint of backward error analysis, an algorithm is stable if the result it produces is the exact solution to a nearby problem
- i.e. the effect of perturbations during the computation is no worse than the effect of a small amount of data error in the input
- A stable algorithm produces exactly the correct result for nearly the correct problem
- Accuracy refers to the closeness of a computed solution to the true solution
- Stability does not by itself guarantee accuracy — accuracy depends on the conditioning of the problem as well as algorithm stability

Stability and Accuracy

- Stability tells us that the solution obtained is exact for a nearby problem — but the solution to that nearby problem is not necessarily close to the solution to the original problem unless the problem is well-conditioned
- Inaccuracy can stem from applying
  - a stable algorithm to an ill-conditioned problem
  - an unstable algorithm to a well-conditioned problem
- Stable algorithm + well-conditioned problem $\Rightarrow$ accurate solution!