Lecture Notes 04: Numerical Algorithms & Errors

CPSC 302: Numerical Computation for Algebraic Problems

Jessica Bosch
jbosch@cs.ubc.ca
http://www.cs.ubc.ca/~jbosch

University of British Columbia
Department of Computer Science

2017/2018 Winter Term 1

Outline

1. Announcements

2. Numerical Algorithms
   - Review
   - Algorithm Properties

Outline

Matlab Tutorial

- MATLAB Tutorial: Monday, here at 6-8pm
- Midterm Exam: October 25, 1-2pm (50 minutes), PHRM 1201
- Upload pre-class quiz today (Connect) and Piazza (waitlist students)

Outline

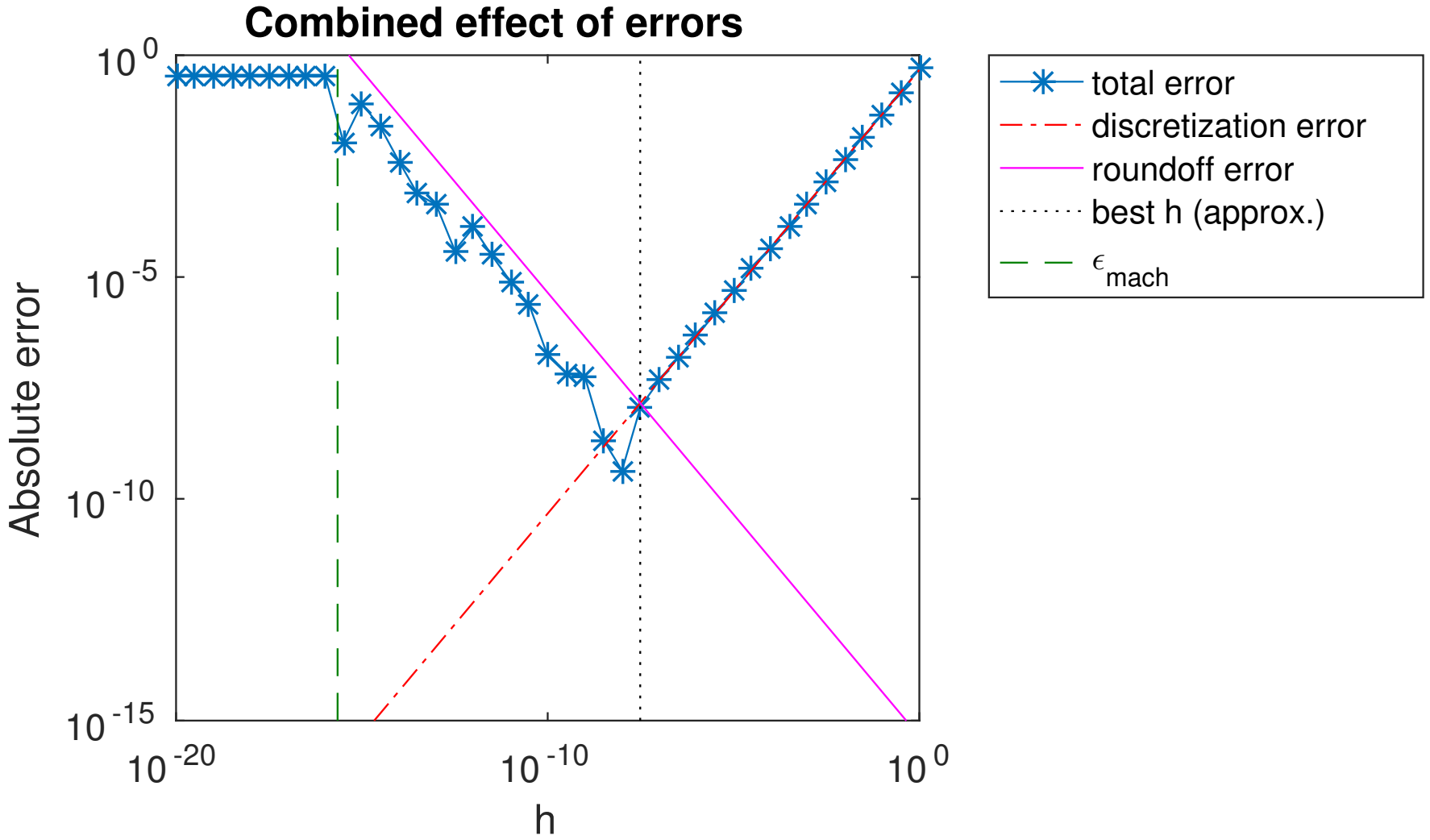
How to Measure Errors

Can measure errors as absolute or relative, or a combination of both.

- The absolute error in v approximating u is $|u - v|$.
- The relative error (assuming $u \neq 0$) is $\frac{|u - v|}{|u|}$.

u	v	Absolute Error	Relative Error
1	0.99	0.01	0.01
10000	9999.99	0.01	0.000001

Combined Effect of Errors



Discretization error: $\left| f'(x_0) - \frac{f(x_0+h)-f(x_0)}{h} \right| \approx \frac{h}{2} |f''(x_0)|$

Outline

1. Announcements

2. Numerical Algorithms
   - Review
   - Algorithm Properties

Algorithm Properties

Remember the good performance features:

1. **Accuracy:**
   When designing numerical algorithms, it is necessary to be able to point out what magnitude of error is to be expected.
   - e.g., $\mathcal{O}(h)$ on the previous slide (derivative approximation)

Algorithm Properties

Remember the good performance features:

Relation to relative error:

- If an approximate value has a relative error of about 10^{-p}, then it approximates the exact value to about p significant digits.
- Example: $u = 0.0275303$, $v = 0.0275194$

  $\rightsquigarrow$ We can see that v has 3 matching significant digits $(2, 7, 5)$.

  $\rightsquigarrow$ In fact, the relative error gives

$$\frac{|u - v|}{|u|} \approx 0.396 \cdot 10^{-3} ,$$

  so $p \approx 3$.

  $\rightsquigarrow$ So here, counting the number of correct significant digits gives us the same value of $p = 3$.

BUT: The concept of significant digits is not simply a matter of counting the number of digits which are correct. See next slide!

Algorithm Properties

Remember the good performance features:

Relation to relative error:

- If an approximate value has a relative error of about 10^{-p}, then it approximates the exact value to about p significant digits.
- Example: $u = 6.000$, $v = 5.999$

  $\rightsquigarrow$ We can see that v has no matching significant digits.

  $\rightsquigarrow$ But, the relative error gives

$$\frac{|u - v|}{|u|} \approx 1.667 \cdot 10^{-4} ,$$

  so $p \approx 4$.

  $\rightsquigarrow$ So here we see that the concept of significant digits is not simply a matter of counting the number of digits which are correct.

Algorithm Properties

Remember the good performance features:

2. **Efficiency:**
   Numerical algorithms should take a reasonable amount of computational time. Depends on CPU time (process time) and storage requirements.

   - usually measured in terms of number of elementary operations (flops - floating point operations)

Algorithm Properties

Remember the good performance features:

3. **Robustness:**
   Ability to cope with errors and to ensure that the algorithm would work under all conditions. There are intrinsic numerical properties that account for the robustness and reliability of an algorithm, i.e., stability.

Counting Flops

How many flops do we need to evaluate a polynomial of degree n

$$p_n(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n$$

at a fixed point x when implementing it in the brute force way as the formula is (without any optimization, rearranging, or temporary storage)?

- $+$: n additions
- $*$: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ multiplications (Gauss formula)

Problem Conditioning and Algorithm Stability

Address the appraisal of a given computed solution.

Qualitatively speaking:

- The problem is ill-conditioned (or sensitive) if a small perturbation in the data may produce a large difference in the result.
  The problem is well-conditioned otherwise.

- The algorithm is stable if its output is the exact result of a slightly perturbed input.
  In other words: The algorithm is stable if the effect of perturbations during computation is no worse than the effect of a small amount of data error in the input for a given problem.

Even if a problem is well-conditioned, it is certainly possible to design unstable algorithms. We will see such an example soon!

Computational and Propagated Data Error

Let us understand the overall effects of approximations in numerical computations. Consider trying to compute $f(x)$ for a given scalar x and function f.

- Due to pre-existing or roundoff errors, we may have $\hat{x}$ rather than x.
- Due to errors during computation, we actually compute with an approximation $\hat{f}$.

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

- Computational error is independent of pre-existing error (i.e., $\hat{x} - x$), so we talk about the algorithm stability.
- Propagated data error is independent of the algorithm (i.e., $\hat{f}$), so we talk about the problem sensitivity.
- In the end, we care about the total error $\hat{f}(\hat{x}) - f(x)$ or accuracy of the result.

Example I

Suppose we have no calculator and need a quick approximation to $\sin(\pi/8)$.

Hence, the true value of the input is $x = \pi/8$, and the desired true result is $f(x) = \sin(\pi/8)$.

We approximate π simply by 3, hence our actual, inexact input is $\hat{x} = 3/8$. We know that for small arguments, a good approximation to $\sin(x)$ is x (this is the first term in the Taylor series expansion). Hence, we work with the truncation $\hat{f}(x) = x$ rather than with the true function $f(x) = \sin(x)$.

Propagated data error introduced by using inexact input:

$$f(\hat{x}) - f(x) = \sin(3/8) - \sin(\pi/8) \approx -0.0164$$

Computational error caused by truncating the infinite series:

$$\hat{f}(\hat{x}) - f(\hat{x}) = 3/8 - \sin(3/8) \approx 0.0087$$

Total error:

$$\hat{f}(\hat{x}) - f(x) \approx -0.0077$$

Example II

In the previous example, the two errors have opposite signs, so they partially offset each other. Also, both errors are roughly similar in magnitude.

In other examples, the two errors may have the same sign and thus reinforce each other. Also, either source of error can dominate.

Problem Sensitivity – (Relative) Condition Number

If the propagated data error is large even when $\hat{x} - x$ is small, then the problem is sensitive or ill-conditioned.

- The condition number of a problem is a quantitative measure of problem sensitivity based on propagated data error

$$\kappa = \frac{\text{relative output error}}{\text{relative input error}} = \frac{|f(\hat{x}) - f(x)|/|f(x)|}{|\hat{x} - x|/|x|}$$

  Notice that if $\hat{x}$ is close to x, then $\kappa \approx \left| \frac{x f'(x)}{f(x)} \right|$.

- The condition number can be thought of as the amplification factor of any input error. Large condition numbers refer to ill-conditioned problems.
- There are many important but ill-conditioned problems: weather, tomography, image deblurring, etc.

Example: Problem Sensitivity

Evaluating $g(x) = \sqrt{x}$ is a well-conditioned problem, because $g'(x) = \frac{1}{2\sqrt{x}}$ and hence $\kappa \approx \left| \frac{xg'(x)}{g(x)} \right| = \frac{1}{2}$. Thus, a small relative change in the input brings about about half of that change in the output. On the other hand, evaluating

$h(x) = \tan(x)$ near $\frac{\pi}{2}$ is an ill-conditioned problem: We have that $h'(x) = \frac{1}{\cos^2(x)}$ in that case, which is very large near $\frac{\pi}{2}$. For instance, setting $x = \frac{\pi}{2} - 0.001$ and $\hat{x} = \frac{\pi}{2} - 0.002$, we have that $|\tan(x) - \tan(\hat{x})| \approx 500.$ Thus, a small perturbation in the input brings about a large perturbation in the output.

Problem Sensitivity – Absolute Condition Number

If the input x or the output $f(x)$ is zero, then the (relative) condition number

$$\kappa = \frac{\text{relative output error}}{\text{relative input error}} = \frac{|f(\hat{x}) - f(x)|/|f(x)|}{|\hat{x} - x|/|x|}$$

is undefined.

Use the absolute condition number instead:

$$\kappa_{abs} = \frac{\text{absolute output error}}{\text{absolute input error}} = \frac{|f(\hat{x}) - f(x)|}{|\hat{x} - x|}$$

Notice that if $\hat{x}$ is close to x, then $\kappa_{abs} \approx \left|f'(x)\right|$.

Algorithm Stability & Overall Accuracy

If the algorithm implementing $\hat{f}$ is designed such that computational errors (roundoff and approximation) do not have a large effect on the output, then the algorithm is stable.

- Warning: "stable" is an overused word in numerical analysis, and has many (and potentially conflicting) meanings.

Overall accuracy — reducing total error — is our goal.

Example: Unstable Algorithm

Problem: Evaluate $f(x) = \mathrm{e}^x$ at $x = -20$.
The problem is well-conditioned since

$$\kappa \approx \left| \frac{x f'(x)}{f(x)} \right| = |x| = 20$$

has a moderate size.

Algorithm: Use the Taylor series expansion

$$\hat{f}(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

See: stability_conditioning.ipynb.

The computed result is $6.1476 \cdot 10^{-9}$. However, the correct result is $2.0612 \cdot 10^{-9}$. So the computed solution has no correct significant digits. The relative error is larger than 1. The algorithm is unstable.

Example: Stable Algorithm for Evaluating e^x at $x = -20$

Since $x < 0$, the terms of the Taylor series alternate in sign. Cancellation errors lead to loss of significant digits.

New algorithm: Make use of the relation $e^{-x} = \frac{1}{e^x}$.

1. Perform the algorithm on the previous slide with $x = +20$. Let us denote the computed solution by $\hat{z}$.
2. Using the above relation, we approximate e^{-x} via $e^{-x} \approx \frac{1}{\hat{z}}$.

The computed result is $2.0612 \cdot 10^{-9}$. Remember, the correct result was $2.0612 \cdot 10^{-9}$. The relative error is of order 10^{-16}.