Lecture Notes 03: 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. Reflection

2. Today's Goals

3. Numerical Algorithms
   - Scientific Computing
   - Numerical Algorithms and Errors

Outline

Matlab Tutorial

- Reaction to the Friday exercise
- For MATLAB newbies
- Time: 18:00-20:00
- Date: I asked for this Wed, Thur or next Mon

In-Class Group Exercises I

Less text/tasks (time flies so fast):

1. I present an example first, and then you do the exercise.
2. You directly start with exercise.

Maybe post exercise in advance

In-Class Group Exercises II

1. Think/Pair/Share:
    - Spend a few minutes on your own, then discuss with partner (or directly start with partner)
    - Two pairs come together (group of 4), discuss in group, and submit the exercise as group
    - Reflection
2. Start in groups of 4: checkpoints during the activity

Outline

1. Reflection

2. Today's Goals

3. Numerical Algorithms

Today's Goals

- Explain what numerical algorithms are.
- Identify sources and types of errors.
- Describe how to measure errors.

Outline

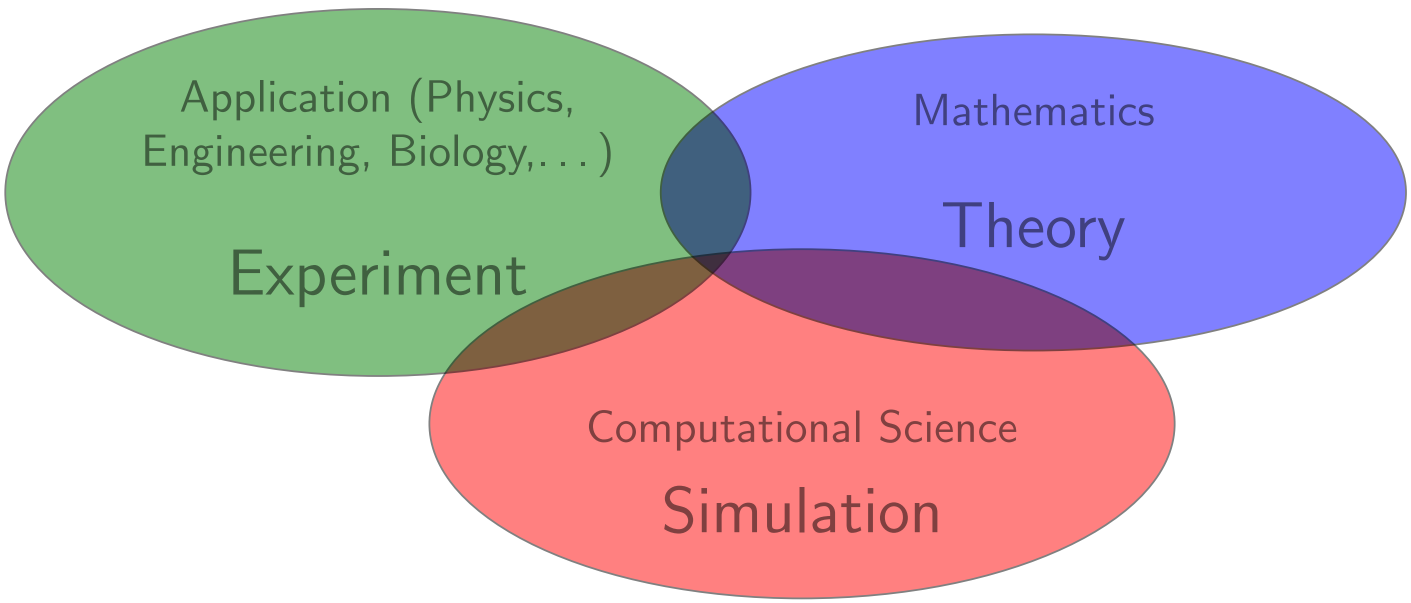
1. Reflection

2. Today's Goals

3. Numerical Algorithms
   Scientific Computing
   Numerical Algorithms and Errors

Scientific Computing

Represent the behavior of a natural process or engineered object so that we can better understand, modify and/or optimize it.



Process of Scientific Computing

Basic steps [Heath, Sec. 1.1.1]:

1. Mathematical Modeling
2. Algorithm
3. Implementation
4. Execution
5. Visualization
6. Validation

Success is measured by: Accuracy, Efficiency, Robustness.

CPSC 302: Focus on steps 2–3 – design, analysis, implementation, and use of numerical algorithms. We assume that an application expert (possibly you) will accomplish steps 1 & 6, and we use MATLAB for steps 4–5.

Example

Consider estimating heat distribution over electric blanket.

1. Mathematical Modeling:
   - Heat Equation: $\frac{\delta u(\mathbf{x},t)}{\delta t} = \Delta u(\mathbf{x},t)$ with boundary conditions

Example

Consider estimating heat distribution over electric blanket.

2. Algorithm:
   - Usually cannot solve partial differential equations (PDEs) analytically for $u(\mathbf{x}, t)$, or even represent a continuous function of a continuous time and space variable
   - Discrete representation for space:
     - Discretize the second spatial derivatives (the Laplace operator Δ)
     - Get a system of ordinary differential equations (ODEs): $\mathbf{y}'(t) = A\mathbf{y}(t)$, where A is a *sparse* matrix
   - Solve system of ODEs:
     - Part of CPSC 303!
     - E.g., implicit methods lead to problems of the form $A\mathbf{z} = \mathbf{b}$ (system of linear equations)
   - Algorithm for solving $A\mathbf{z} = \mathbf{b}$? $\rightarrow$ CPSC 302!
     - For us, this would be the starting point.

Example

Consider estimating heat distribution over electric blanket.

3. Implementation:
   - We need to write a function

$$\texttt{uf} = \texttt{solveHeat}(\ldots)$$

   - How should we choose time and space discretization parameter to achieve some particular level of error?

Example

Consider estimating heat distribution over electric blanket.

4. Execution:
   - What machine should we use?

Example

Consider estimating heat distribution over electric blanket.

5. Visualization:
    - 3D plot, video?
    - Photo-realistic images?

Example

Consider estimating heat distribution over electric blanket.

6. Validation:
   - How do we check whether our simulation is reasonable?
     - E.g., compare with physical experiments.
   - If it is not accurate enough, how can we improve it?
     - Better measurement of actual initial temperature?
     - Better mathematical model?
     - Higher resolution (smaller time step size and and spatial grid size)?
     - Better algorithm for solving system of linear equations?

Typical Scientific Computing Strategies

We have complicated, continuous and/or infinite problems which we seek to solve on a computer with simple operations, discrete data types and finite memory.

- real numbers $\rightsquigarrow$ **fixed number of digits**
- functions $\rightsquigarrow$ **samples**
- differential equations $\rightsquigarrow$ **algebraic equations**
- nonlinear problem $\rightsquigarrow$ **linear problem**
- general matrices $\rightsquigarrow$ **simpler matrices**

We do **not** usually have exact replacements, so we are forced to use approximations and hence deal with error.

Sources of Error

Pre-existing errors:

- **Modeling**: equations do not match the physical process (simplification/omission of, e.g., friction)
- **Measurement**: input data not known exactly
- **Previous computations**: data and/or equations come from some previous computation which was not exact

Optimist's view: There is no reason to struggle to make our algorithm or implementation more accurate than the model and data.

Sources of Error

Errors during computation:

- Approximation errors
  - Discretization errors: discretizations of continuous processes, e.g., interpolation, numerical differentiation & integration
  - Convergence errors: iterative methods (cut after a finite number of iterations)
- Roundoff errors: finite precision arithmetic (floating point numbers are not the same as real numbers)

Dealing with Error

Can not avoid error, but must try to avoid allowing it to compound.

In **CPSC 302**, we will spend

- no time on pre-existing error,
- the rest of the term on approximation and roundoff errors.

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
1	1.01	0.01	0.01
-1.5	-1.2	0.3	0.2
100	99.99	0.01	0.0001
100	99	1	0.01

Example

Given smooth function $f(x)$, approximate derivative at some point $x = x_0$:

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h},$$

for a small parameter value h.

Discretization error:

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

Results

Discretization error:

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

Try for $f(x) = \sin(x)$ at $x_0 = 1.2$.
(So we are approximating $f'(1.2) = \cos(1.2) = 0.362357754476674...$.)

h	Absolute error
0.1	4.716676e-2
0.01	4.666196e-3
0.001	4.660799e-4
1.e-4	4.660256e-5
1.e-7	4.619326e-8

These results reflect the discretization error as expected, since:

$$\frac{h}{2} \left| f''(x_0) \right| \approx -0.466\, h$$

Results for Smaller h

Making h smaller, can we achieve an arbitrary accuracy, e.g.,

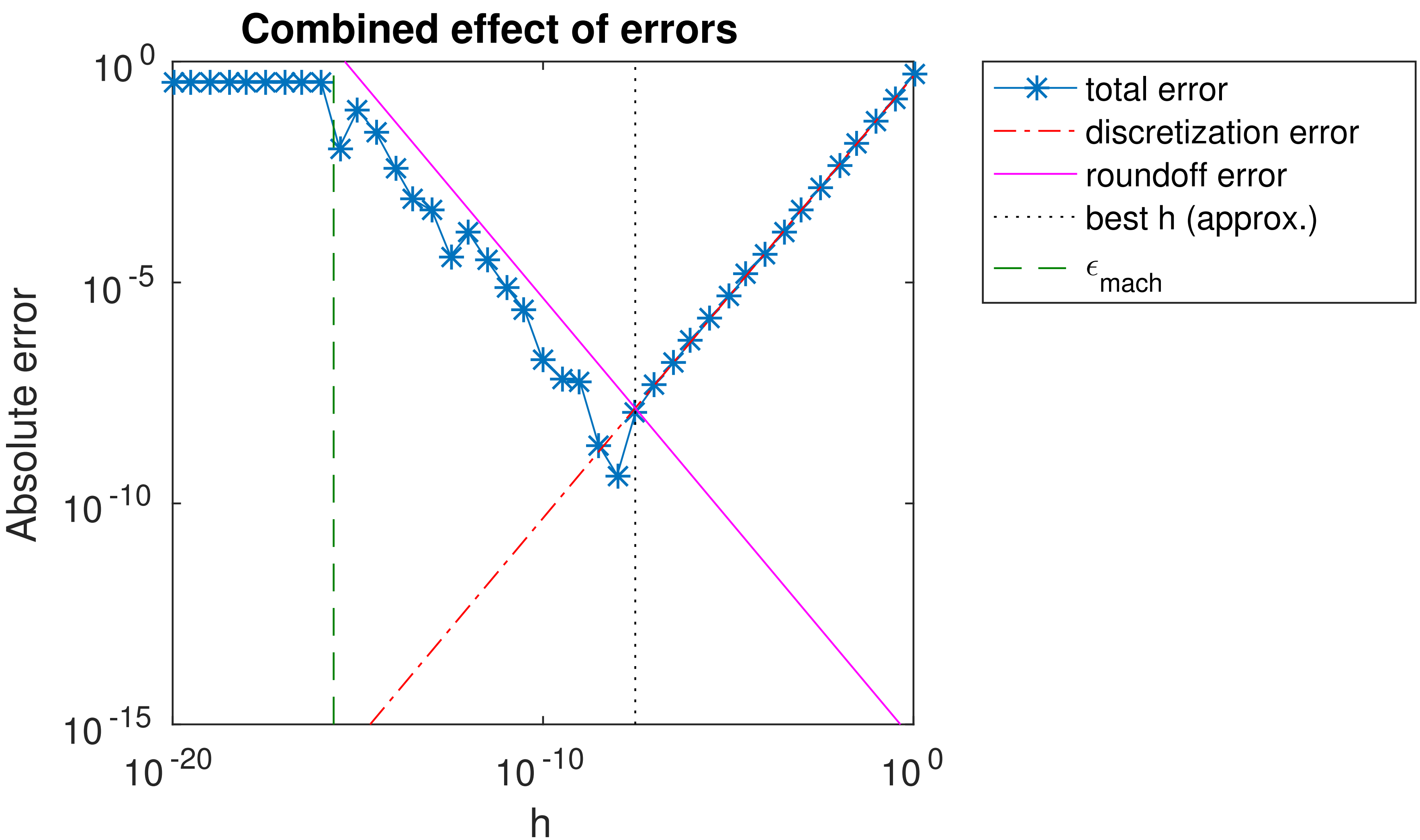
$$\left| f'(x_0) - \frac{f(x_0+h) - f(x_0)}{h} \right| < 10^{-10} \ ?$$

h	Absolute error
1.e-8	4.361050e-10
1.e-9	5.594726e-8
1.e-10	1.669696e-7
1.e-11	7.938531e-6
1.e-13	6.851746e-4
1.e-15	8.173146e-2
1.e-16	3.623578e-1

These results reflect both discretization and roundoff errors.

See: combined_error.ipynb

Results for All h



$\left| f'(x_0) - \frac{f(x_0+h)-f(x_0)}{h} \right|$, $\frac{h}{2}\,|f''(x_0)|$, $\frac{2\epsilon_{\text{mach}}}{h}$, $2\sqrt{\frac{\epsilon_{mach}}{|f''(x_0)|}}$, $\epsilon_{mach} \approx 2.2 \cdot 10^{-16}$