Lecture Notes 0: Introduction

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

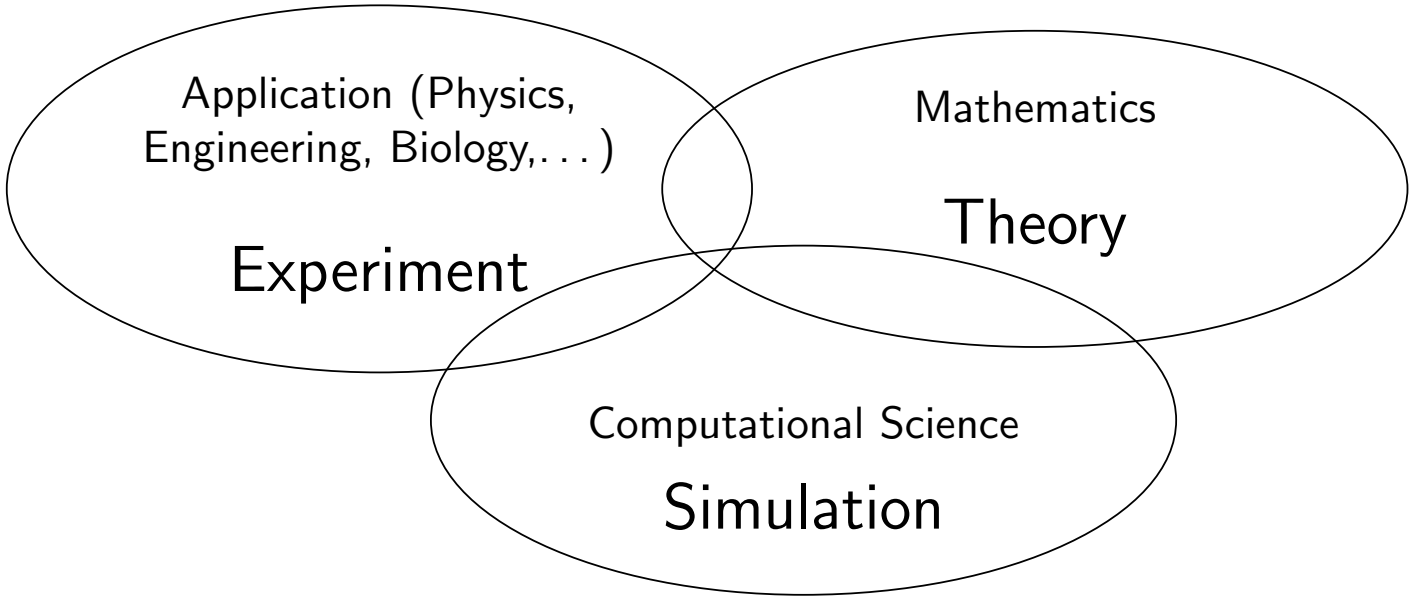
Outline

1. Motivation
2. Introduction of Your Instructor
3. Syllabus
4. Group Activity
   - Common Ground
   - Team Presentation
5. Reflection

Outline

Scientific Computing

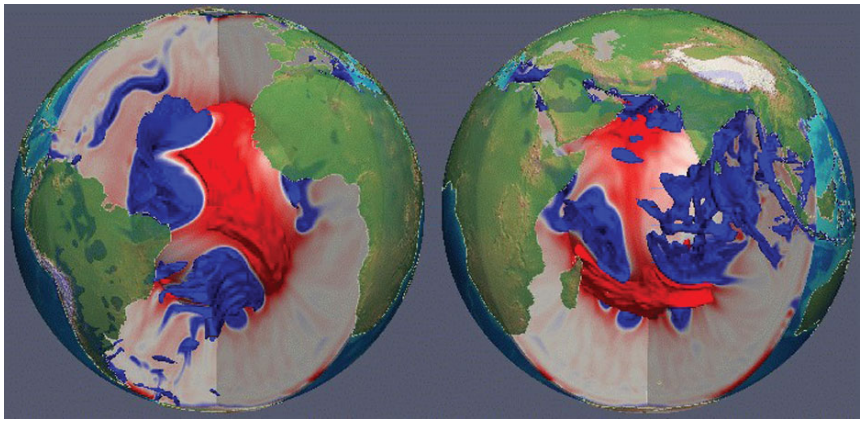
- Development & study of numerical algorithms for solving mathematical problems.
- Numerical analysis is the theory behind.



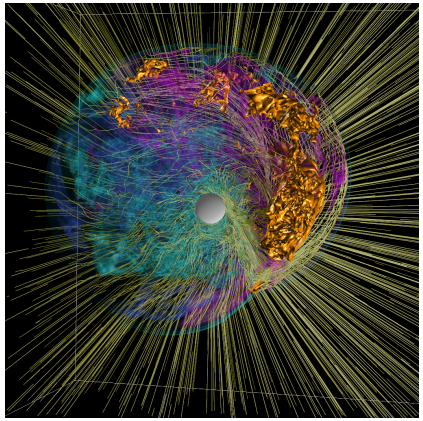
Why Numerical Simulations?

Experiments are often impossible.

- Geophysics: thermal structure of the Earth's mantle [1]

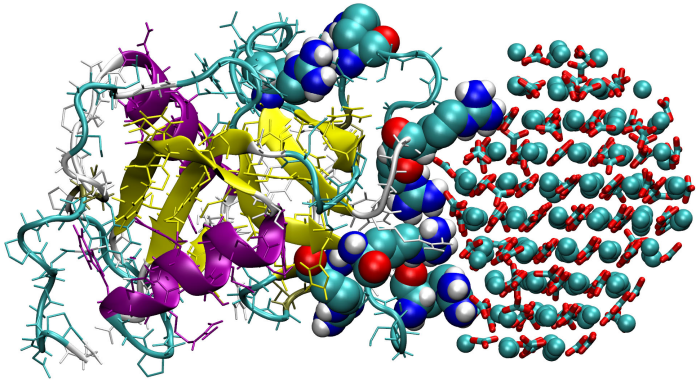


- Astrophysics: supernova simulations [2]

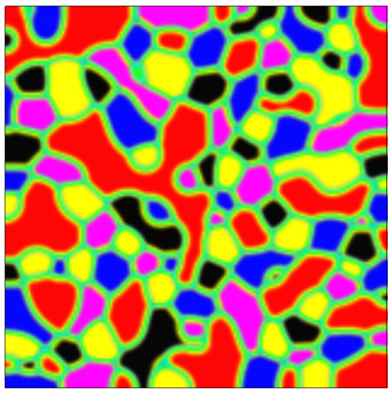


Why Numerical Simulations?

- Experiments are often costly: analysis and study of proteins [3]



- Simulations are often cheaper and faster: phase separation and coarsening in alloys [4]



Why Numerical Methods?

- Practical applications often require solving enormous systems of equations, millions or even billions of variables.

- Many practical problems – optimization, differential equations, signal processing, etc. – boil down to solving linear systems, even when the original problems are non-linear. Finite element software, for example, spends nearly all its time solving linear equations.

- The heart of Google is an enormous linear algebra problem. PageRank is essentially an eigenvalue problem.

To Prevent Disasters

Collapse of the Tacoma Narrows bridge on November 7, 1940

`http://eigenvaluesandbridges.weebly.com/tacoma-narrows-bridge.html`

A suspension bridge should be built so that the lower eigenvalues of vibrations in the bridge are not close to possible wind-induced frequencies.

CPSC 302

- We study **basic numerical methods** for solving:
  - Linear systems
  - Linear eigenvalue problems
  - Linear least squares problems
  - Nonlinear equations

- Many of the algorithms we will learn are already implemented...
- Why studying them?
  - They provide the **basis for more complex tasks**.
  - To **detect, understand, and correct errors**.
  - To **modify or create software** appropriate to a particular problem.
  - To **formulate the problem** in a manner appropriate to the problem.

Outline

My Home


Sign in
Germany
Poland
Netherlands
Belgium
Czechia
Slovakia
Austria
Hungary
Switzerland
Liechtenstein
Amsterdam
The Hague
Hamburg
Bremen
Berlin
Hanover
Wolfsburg
Brunswick
Magdeburg
Dortmund
Essen
Cologne
Antwerp
Brussels
Lille
Leipzig
Dresden
Frankfurt
Mannheim
Karlsruhe
Stuttgart
Nuremberg
Augsburg
Munich
Strasbourg
Zürich
Salzburg
Graz
Vienna
Bratislava
Budapest
Debrecen
Prague
Brno
Ostrava
Katowice
Kraków
Rzeszow
Lviv
Львів
Ternopil
Тернопіль
Ivano-Frankivsk
Івано-Франківськ
Košice
Zakopane
Wrocław
Łódź
Poznań
Warsaw
Lublin
Kielce
Brest
Брэст
Grodno
Гродна
Białystok
Olsztyn
Gdańsk
Gdynia
Kaliningrad
Калининград
Vilnius
Szczecin
Bydgoszcz
Rostock
Lübeck
Kiel
Odense
Paris
Tours
Oxford
London
Cambridge
Brighton
Satellite
Google

Today's Goals

- Start the term on a positive note
- Gain your interest
- Communicate key elements and expectations for the course
- Actively involve you

Another Passion

Outline

Contact Your Instructor

How to reach me?

- In person after class
- Office hour: Mondays 4:30–5:30pm in ICCS 215
- Piazza
- Email: jbosch@cs.ubc.ca

Contact Your TAs

- Michael Wathen
  - Office hour: Tuesdays 09:30-10:30 in ICCS X337
  - Piazza
  - Email: mwathen@cs.ubc.ca
- Nicholas Hu
  - Office hour: Tuesdays 11:30-12:30 in Demco Learning Centre (ICCS X150), Table 3
  - Piazza
  - Email: nicholas.hu@alumni.ubc.ca
- Sarah Elhammadi
  - Office hour: Wednesdays 11:30-12:30 in Demco Learning Centre (ICCS X150), Table 3 (start: Sept 20)
  - Piazza
  - Email: shammadi@cs.ubc.ca

Outcomes

Main Objective

By the end of this course, you should be able to
choose, analyze, implement, and evaluate
appropriate numerical methods for solving a variety of mathematical problems.

Further outcomes include:

- Improve or continue on being an intentional learner.
- Identify the interactions between numerical computation and other realms of knowledge such as history, economics, science, ...
- Get excited!

Course Structure

	Monday		Wednesday		Friday	
In-class activities		Lecture		Lecture		Group Exercise
Out-of-class activities	Prior Knowledge Quiz		Assignment		Pre-Class Readings Quiz	

Lectures

- we'll use clickers $\rightsquigarrow$ register your clicker on Connect
- receive an extra 1% after good performance

Required Reading

- check "Schedule" on course website regularly
- for Friday classes: complete readings before class
- pre-class quizzes (Connect) on these readings
- in-class group exercises on these readings

Assignments

- bi-weekly
- uploaded ≈ 2 weeks before the due date
- hardcopy and electronic submission
- 1 bonus assignment at the end, up to an additional 3%

Course grade:

- 30% – Assignments
- 5% – Prior Knowledge Quizzes
- 5% – Pre-Class Readings Quizzes
- 5% – In-Class Group Exercises
- 15% – Midterm Exam
- 40% – Final Exam

You must have a pass standing in the assignment component of the course.

Course Structure

	Monday		Wednesday		Friday	
In-class activities		Lecture		Lecture		Group Exercise
Out-of-class activities	Prior Knowledge Quiz		Assignment		Pre-Class Readings Quiz	

Prior Knowledge Quizzes

- in Connect
- weekly, due by Monday at 1pm
- 5 graded quizzes among them

Pre-Class Readings Quizzes

- in Connect
- weekly, due by Friday at 1pm
- 5 graded quizzes among them

In-Class Group Exercises

- weekly
- 6 graded exercises among them

Course grade:

- 30% – Assignments
- 5% – Prior Knowledge Quizzes
- 5% – Pre-Class Readings Quizzes
- 5% – In-Class Group Exercises
- 15% – Midterm Exam
- 40% – Final Exam

You must have a pass standing in the assignment component of the course.

Suggestion Box

You can provide me with
anonymous suggestions regarding my teaching or the course in general.

You are encouraged to submit *typed* suggestions if you are concerned.

Course Details: Getting Started

- **Find the course website:**
  `http://www.cs.ubc.ca/~jbosch/courses/2017-18/CPSC302/`
- Read through the syllabus on the first page.
- Log into Connect and register your i>clicker.
  Clicker bonus questions start Sept 20.
- Register into Piazza site for the course:
  `http://www.piazza.com/ubc.ca/winterterm12017/cpsc302/`
- Get access to MATLAB.
- Do the assigned reading for Friday and answer the pre-class reading quiz.
  This Friday, each group should bring a laptop with MATLAB access.
- Interest Inventory Survey (Connect) due by Monday, 1pm

Note for Waiting List Students

- I will copy a PDF of the quizzes/surveys into Piazza until Sept 19.
- Students are admitted from wait lists until the add/drop deadline according to the department's standard priority scheme: `https://www.cs.ubc.ca/students/undergrad/courses/waitlists`

Outline

Common Ground

Form groups of 4 students.

Common Ground

Form groups of 4 students.

Discuss and write down the following on a piece of paper:

- Everyone's name
- Come up with 6 things that you all share
- Create a team name
- Try to find a volunteer that is keen to present your team to the class
- If you have time: choose a team song, design a team mascot

Team Presentations

Who wants to present the team to the whole class?

Outline

Reflection

On a piece of paper, write down the following:

- Your name
- The best way to contact you
- Answer the following question:

  What have teachers and professors done in the past that helped you to learn?

Submit the paper to your instructor.

References

[1] "Thermal versus elastic heterogeneity in high-resolution mantle circulation models with pyrolite composition: High plume excess temperatures in the lowermost mantle" by Schuberth, B. S. A.; Bunge, H. P.; Steinle-Neumann, G.; et al Geochemistry Geophysics Geosystems 10(1) Article Q01W01; Published January 8, 2009.

[2] `http://www.csm.ornl.gov/astro/`

[3] `http://www2.warwick.ac.uk/newsandevents/pressreleases/researchers_apply_computing/`

[4] "Fast Iterative Solvers for Cahn–Hilliard Problems" by J. Bosch; PhD thesis, Faculty of Mathematics, Otto-von-Guericke-Universität Magdeburg, June 2016.

Group Exercise 1: Introduction to Matlab

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

Outline

Midterm Exam

October 25, 13:00-14:00, in PHRM 1201

Contact Your TAs

- Michael Wathen
  - Office hour: Tuesdays 09:30-10:30 in ICCS X337
  - Piazza
  - Email: mwathen@cs.ubc.ca
- Nicholas Hu
  - Office hour: Tuesdays 11:30-12:30 in Demco Learning Centre (ICCS X150), Table 3
  - Piazza
  - Email: nicholas.hu@alumni.ubc.ca
- Sarah Elhammadi
  - Office hour: Wednesdays 11:30-12:30 in Demco Learning Centre (ICCS X150), Table 3 (start: Sept 20)
  - Piazza
  - Email: shammadi@cs.ubc.ca

Outline

Today's Goals

- Apply knowledge of pre-reading
- Get a feel for MATLAB
- Interact with other students in a team

Outline

Introduction to Matlab

Form groups of 4–5 students.

1. Reflection of Quiz: Group Discussion
2. Open Questions?
3. MATLAB Group Exercise
4. Hardcopy and electronic submission

Outline

Reflection

On a piece of paper, answer the following question:

How useful was the group exercise that we did in class today?

Put the paper into the "Suggestion Box" or next to it.

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

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

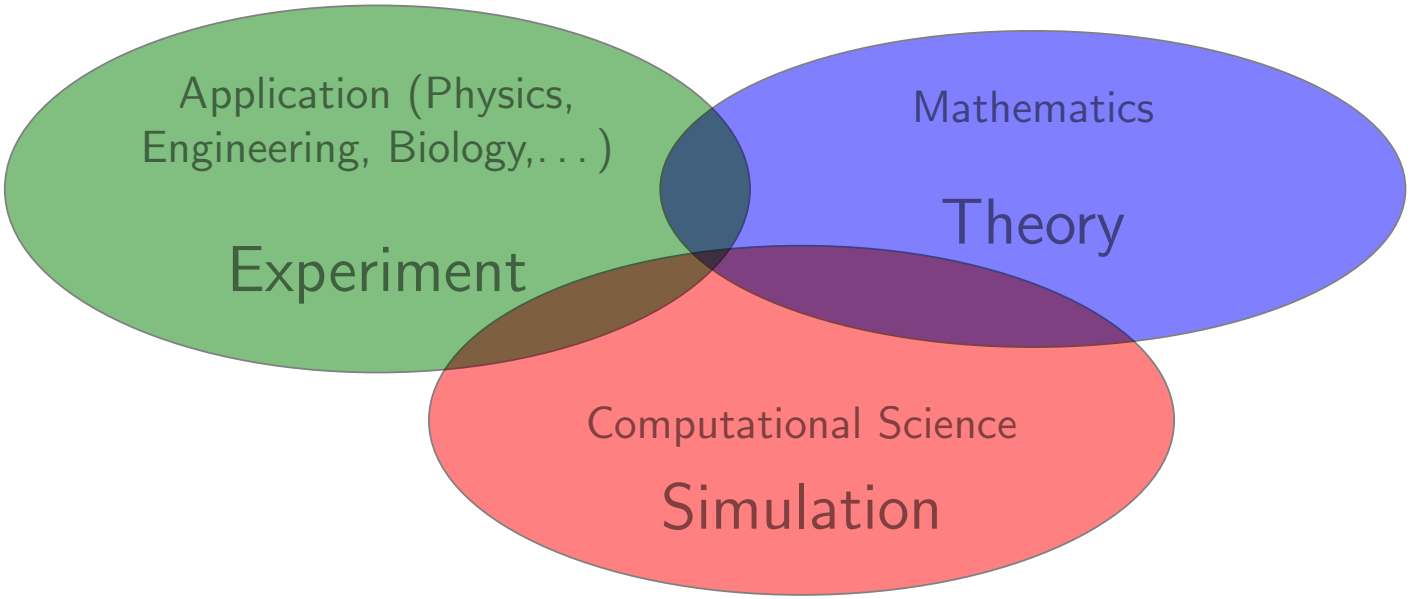
Today's Goals

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

Outline

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.,

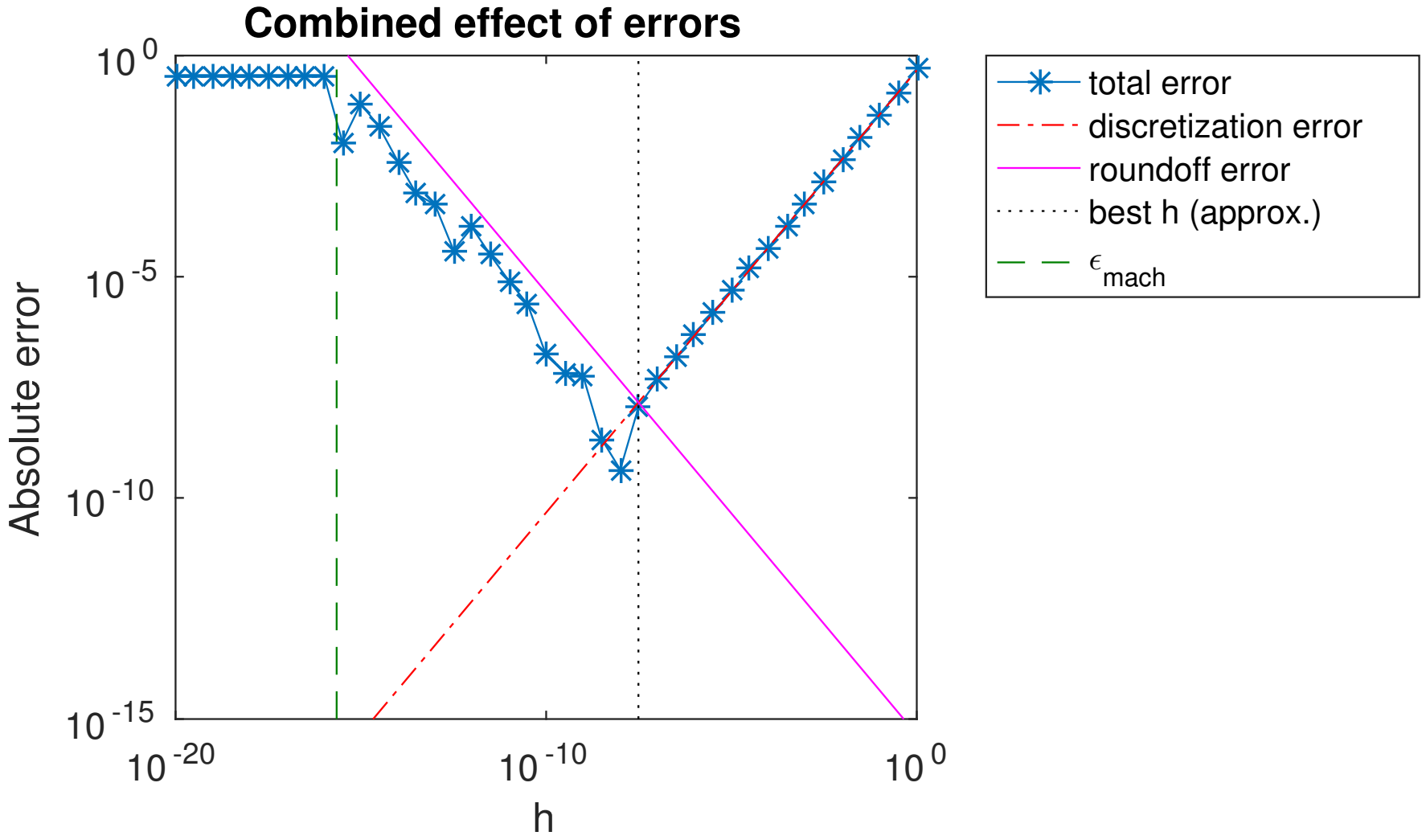
$$\left| f'(x_0) - \frac{f(x_0 + h) - f(x_0)}{h} \right| < 10^{-10} \ ?$$

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



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

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

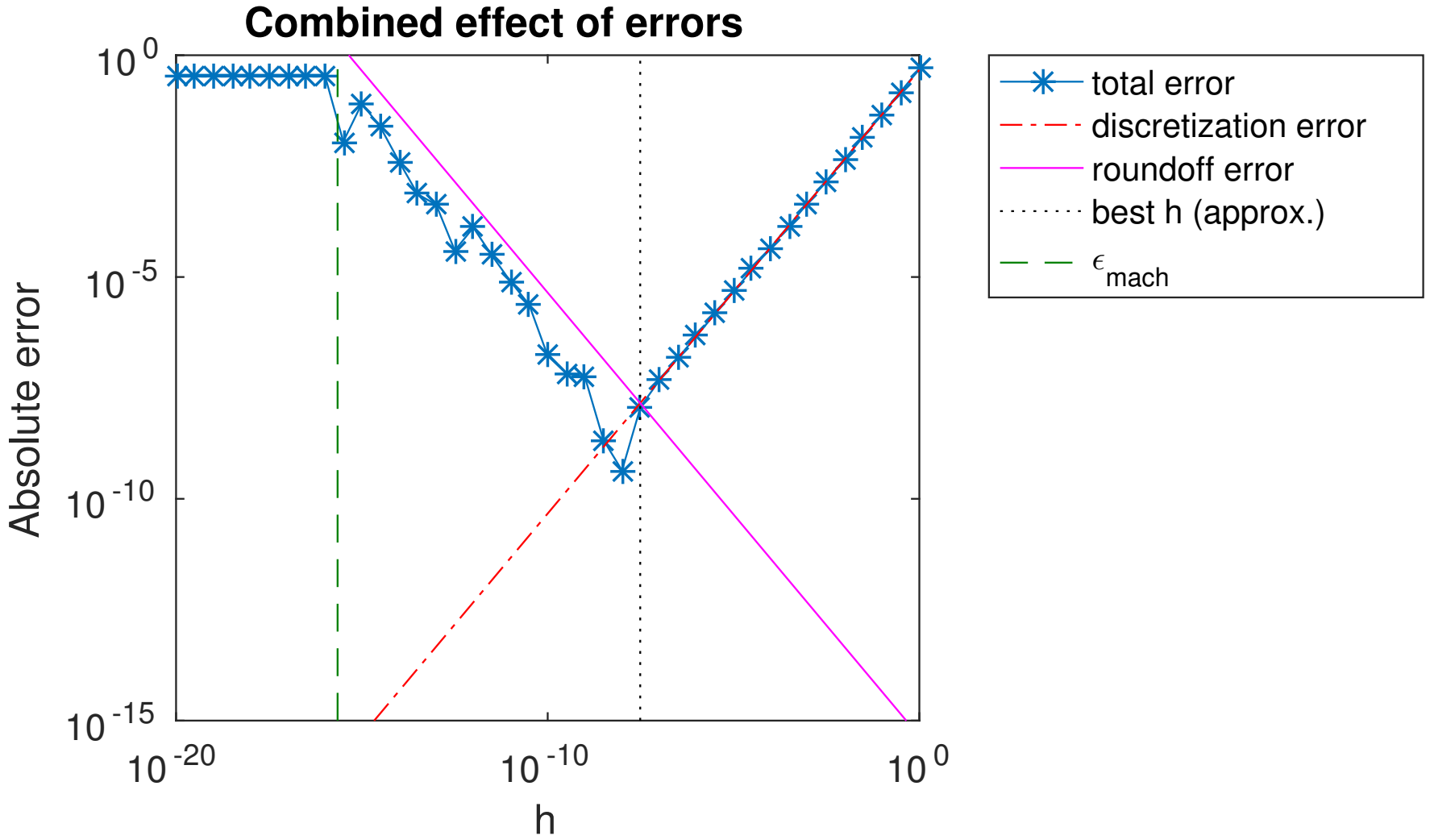
How to Measure Errors

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}.

Group Exercise 2: Algorithm Stability

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. Announcement
2. Outcome
3. Example
4. Group Exercise 2
5. Reflection

Outline

Matlab Tutorial

- MATLAB Tutorial: Monday, here at 6-8pm
- There will be exercises so bring your laptop and have access to MATLAB.
- You can also work with a partner.

Outline

Today's Goal

To illustrate the potentially damaging effect of roundoff errors.

Outline

Error Accumulation I

Consider the problem of calculating the integral

$$u_n = \int_0^1 \frac{x^n}{x+3}\, dx,$$

for the values $n = 0, 1, 2, \ldots$.

1. Derive a formula expressing u_n in terms of u_{n-1} exactly: Find the scalars a and b such that
$$u_n = a * u_{n-1} + b\,.$$
2. Stable?

Error Accumulation II

We obtain the recursive formula $u_n = -3u_{n-1} + \frac{1}{n}$.

$\rightsquigarrow$ Magnitude of roundoff errors gets multiplied by 3 each time.

$\rightsquigarrow$ Exponential error growth.

Roundoff Error Accumulation

- In general, if E_n is error after n elementary operations, cannot avoid linear roundoff error accumulation

$$E_n \simeq c_0 n E_0.$$

- Will not tolerate an **exponential** error growth such as

$$E_n \simeq c_1^n E_0 \text{ for some constant } c_1 > 1$$

  – an **unstable algorithm**.

Outline

Important

- Groups of 2 up to 4 students

  You choose the size – some may prefer smaller groups, some don't.

- We will **not** grade any exercise that has only 1 name on it or more than 4 names.

- **Submit the exercise sheet at the end of today's class directly to your TAs.**

  Any delay will not be accepted.

Outline

Reflection

On a piece of paper, answer the following question:

> *How useful was the group exercise that we did in class today, compared to the last group exercise?*

Put the paper into the "Suggestion Box" or just on the table.

Lecture Notes 06: 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

Outline

Goals

- Describe how numbers are stored in a floating point system.
- Understand how standard floating point systems are designed and implemented: IEEE standard.

Real Number Representation: Decimal & Binary

The decimal system is convenient for humans.

$$(0.0007396)_{10} = (7.396)_{10} \cdot 10^{-4} = \left(\frac{7}{10^0} + \frac{3}{10^1} + \frac{9}{10^2} + \frac{6}{10^3} \right) \times 10^{-4}$$

But computers prefer binary systems.

$$\begin{aligned}
(101.001)_2 = (1.01001)_2 \cdot 2^2 &= \left(\frac{1}{2^0} + \frac{0}{2^1} + \frac{1}{2^2} + \frac{0}{2^3} + \frac{0}{2^4} + \frac{1}{2^5} \right) \times 2^2 \\
&= \left(\frac{1}{2^0} + \frac{1}{2^2} + \frac{1}{2^5} \right) \times 2^2 \\
&= \frac{2^2}{2^0} + \frac{2^2}{2^2} + \frac{2^2}{2^5} \\
&= 4 + 1 + 0.125 = (5.125)_{10}
\end{aligned}$$

Floating Point Number System $\mathbb{F}$

$$\mathbb{F} = \mathbb{F}(\beta, t, L, U)$$

$$x \in \mathbb{F}: \qquad x = \pm \left(\frac{\tilde{d}_0}{\beta^0} + \frac{\tilde{d}_1}{\beta^1} + \frac{\tilde{d}_2}{\beta^2} + \cdots + \frac{\tilde{d}_{t-1}}{\beta^{t-1}} \right) \times \beta^e$$

$$= \pm \left(\tilde{d}_0 . \tilde{d}_1 \tilde{d}_2 \cdots \tilde{d}_{t-1} \right)_\beta \times \beta^e$$

$$x = (\text{sign})(\text{"mantissa", i.e., significant digits})_\beta \times (\text{base}^{\text{exponent}})$$

- Base of the number system: $\beta \in \mathbb{N}$, $\beta > 1$
- Precision, i.e., number of digits with which a number is expressed: t
- Exponent range: $e \in \mathbb{Z}$ with $L \leq e \leq U$
- Digits: $\tilde{d}_i \in \mathbb{N}_0$, $0 \leq \tilde{d}_i \leq \beta - 1$

  Note that $\tilde{d}_0 > 0$: normalized floating point representation

Standard Floating Point Number System

Typical floating point number systems:

type	β	t	e
IEEE standard single precision (C float)	2	24	[–126, +127]
IEEE standard double precision (C double, MATLAB)	2	53	[–1022, +1023]

Lots of technical details: normalization ($\tilde{d}_0 \neq 0$), rounding, subnormals and gradual underflow, cancellation, special values (0, `Inf`, `NaN`), etc.

IEEE Standard Word

Sign, exponent, and mantissa stored in separate fields of a given floating point word.

Double precision (64 bit word)

$s = \pm$	$b =$11-bit exponent	$f =$52-bit "mantissa"
$s \in \{0, 1\}$	$b = e + U$ in binary representation	$f = \tilde{d}_1 \tilde{d}_2 \cdots \tilde{d}_{t-1}$

Single precision (32 bit word)

$s = \pm$	$b =$8-bit exponent	$f =$23-bit "mantissa"

Example

How do we store $(1.01101)_2 \times 2^4$ as IEEE single precision word?

s	b	f
0	10000011	01101000000000000000000

Rounding

How do we approximate an $x \in \mathbb{R}$ for which $x \notin \mathbb{F}$?

Example: Consider $\mathbb{F}(10, 4, -2, 2)$ and $x = \left(\frac{8}{3}\right)_{10}$. Obviously, $x \notin \mathbb{F}$.

$$\left(\frac{8}{3}\right)_{10} = 2.66\ldots6 \times 10^0 = \underbrace{\left(\frac{2}{10^0} + \frac{6}{10^1} + \frac{6}{10^2} + \frac{6}{10^3} + \frac{6}{10^4} + \ldots\right)}_{\text{in } \mathbb{F}\text{: only 4 significant digits}} \times 10^0$$

Rounding: $\text{fl}(x) \approx x$ where $\text{fl}(x) \in \mathbb{F}$

- *Chop:* truncate after the 3rd digit

$$x = \left(\frac{8}{3}\right)_{10} \approx \quad \text{fl}(x) = \left(\frac{2}{10^0} + \frac{6}{10^1} + \frac{6}{10^2} + \frac{6}{10^3}\right) \times 10^0 = 2.666 \times 10^0$$

- *Round to nearest:* default rule in IEEE standard systems

$$x = \left(\frac{8}{3}\right)_{10} \approx \quad \text{fl}(x) = \left(\frac{2}{10^0} + \frac{6}{10^1} + \frac{6}{10^2} + \frac{7}{10^3}\right) \times 10^0 = 2.667 \times 10^0$$

Examples

Number	Chop	Round to nearest
1.49	1.4	1.5
1.51	1.5	1.5
1.99	1.9	2.0
1.55	1.5	1.6
1.45	1.4	1.4

Machine Precision ϵ_{mach}

Accuracy of $\mathbb{F}$ is measured by the machine precision ϵ_{mach}.

- It is the maximal possible relative error in representing an $x \neq 0$ in $\mathbb{F}$:

$$\left| \frac{\mathsf{fl}(x) - x}{x} \right| \leq \epsilon_{mach}$$

- Its actual value depends on rounding rule:
  - Chop: $\epsilon_{mach} = \beta^{1-t}$
  - Round to nearest: $\epsilon_{mach} = \frac{1}{2}\beta^{1-t}$

- IEEE standard single precision: $\epsilon_{mach} = 2^{-24} \approx 10^{-7}$
- IEEE standard double precision: $\epsilon_{mach} = 2^{-53} \approx 10^{-16}$

  MATLAB: ϵ_{mach}=eps/2

Floating Point Arithmetic & Rounding

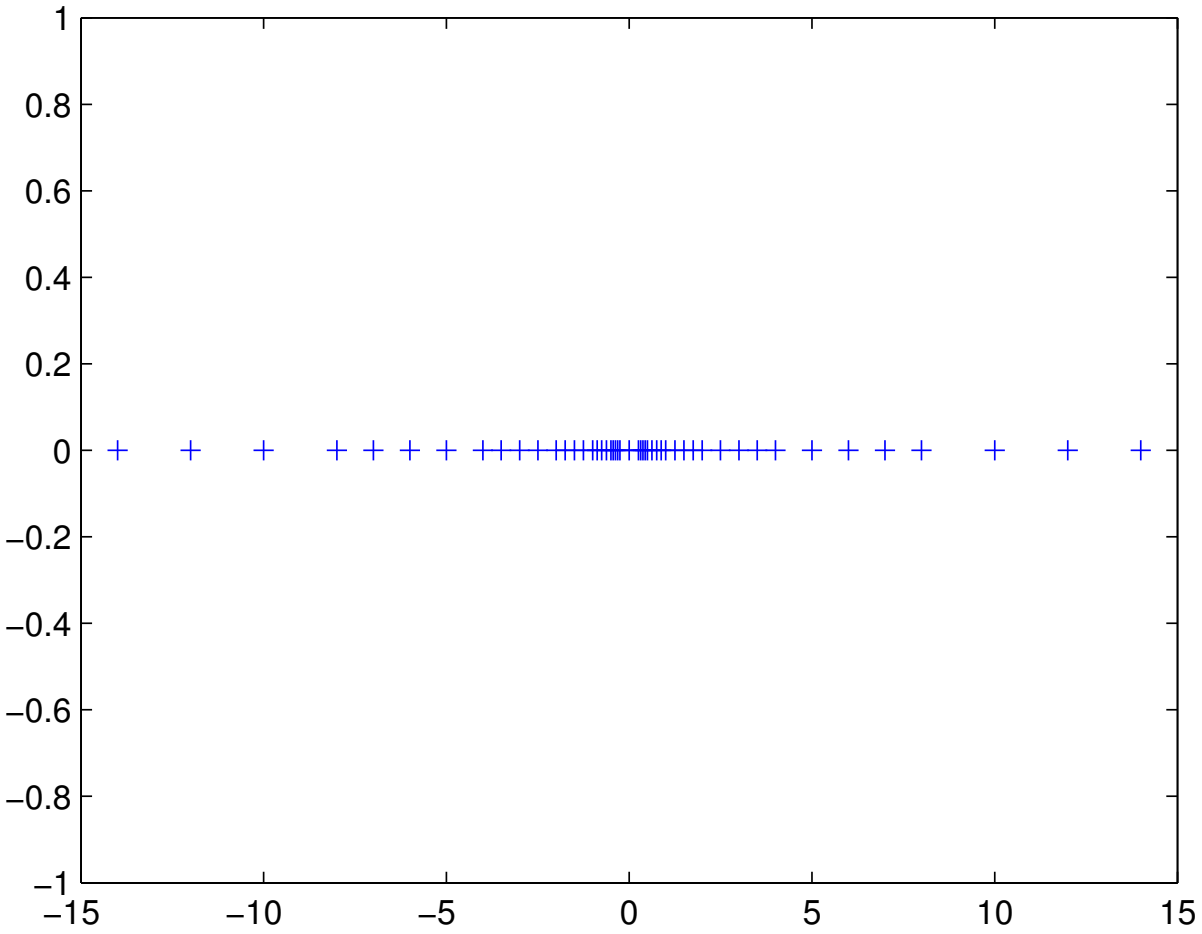
The result of an arithmetic operation on floating point numbers may not be a floating point number.

- Arithmetic operations are commutative (eg: $a + b = b + a$).
- Arithmetic operations are **not** associative (eg: $(a + b) + c \neq a + (b + c)$ for some a, b, c).
  - See Piazza: roundoff_error.ipynb
- For error analysis, usually assume (true for IEEE standard)

$$\frac{|\text{fl}(\text{fl}(x) \text{ op } \text{fl}(y)) - (\text{fl}(x) \text{ op } \text{fl}(y))|}{|(\text{fl}(x) \text{ op } \text{fl}(y))|} = |\delta| \leq \epsilon_{mach}$$

  where "op" is any standard arithmetic operation (eg: $+$, $-$, $\times$, :).

Spacing of Floating Point Numbers



Note the **uneven** distribution, both for large exponents and near 0.

Overflow, Underflow, NaN

- Overflow: when $e > U$ (fatal)
- Underflow: when $e < L$ (non-fatal: set to 0 by default)
- NaN: Not-a-number (e.g., $0/0$)

The Patriot Missile Failure

Poor handling of roundoff errors causes disaster!

1991, during the Gulf War, an American Patriot Missile battery in Dhahran, Saudi Arabia, failed to track and intercept an incoming Iraqi Scud missile.

- Chopping of $1/10$ lead to an error of 0.34 seconds.
- Incoming Scud was outside the "range gate" that the Patriot tracked.
- It killed 28 Americans.

Lecture Notes 07: 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

Outline

Waitlist Problem

SOLVED!!!

We are 81 students!

Outline

Goal

Identify & avoid different sources of roundoff error growth.

Roundoff Errors

- Roundoff error is generally inevitable in numerical algorithms involving real numbers.
- People often like to pretend they work with exact real numbers, ignoring roundoff errors, which may allow concentration on other algorithmic aspects.
- However, carelessness may lead to disaster!

  Example: Patriot missile failure in 1991

Roundoff Error Accumulation

- In general, if E_n is error after n elementary operations, cannot avoid linear roundoff error accumulation

$$E_n \simeq c_0 n E_0.$$

- Will not tolerate an **exponential** error growth such as

$$E_n \simeq c_1^n E_0 \text{ for some constant } c_1 > 1$$

  – an **unstable algorithm**.

- Example: recursive formula $y_n = \frac{1}{n} - 100 y_{n-1}$

  $\rightsquigarrow$ magnitude of roundoff errors gets multiplied by 100 each time

Roundoff Error Accumulation

When we design an algorithm, we must **keep error accumulation in mind**:

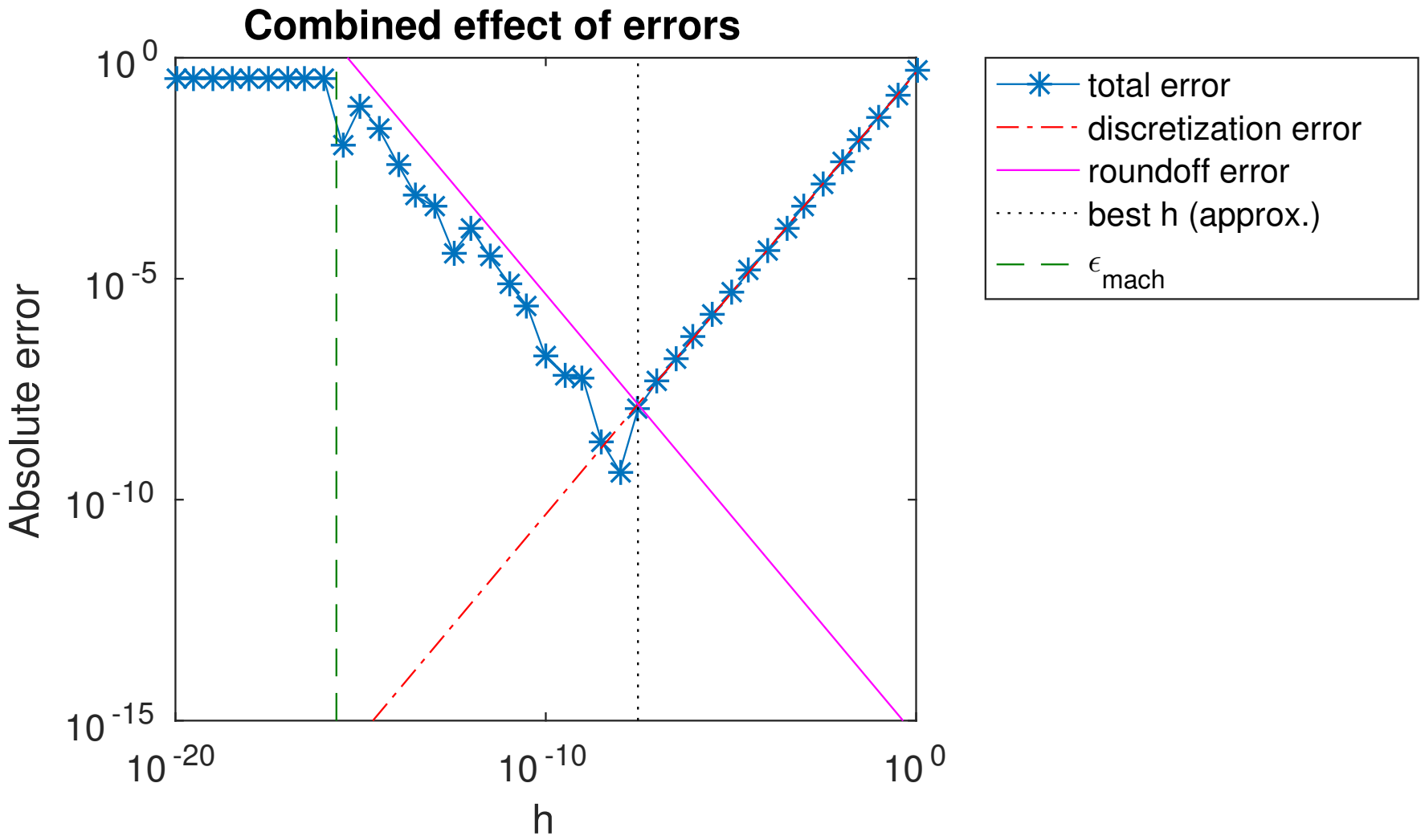
The computer may run millions of operations, and what may look like a meaningless error may quickly have a disastrous effect.

Cancellation Error

When two nearby numbers are subtracted, the relative error is large. That is, if $x \simeq y$, then $x - y$ has a large relative error.

This occurs in practice consistently and naturally, as we will see.

Review Example



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

Review Example

Function evaluation at nearby arguments:

- If $g(\cdot)$ is a smooth function then $g(t)$ and $g(t+h)$ are close for h small.
- But rounding errors in $g(t)$ and $g(t+h)$ are unrelated, so they can be of opposing signs!
- For numerical differentiation, e.g.

$$f'(x_0) \approx \frac{f(x_0+h) - f(x_0)}{h}, \quad 0 < h \ll 1,$$

  if the relative rounding error in the representation is bounded by ε_{mach} then in $|g(t+h) - g(t)|/h$ it is bounded by $2\,\varepsilon_{mach}/h$. This (tight) bound is much larger than ε_{mach} when h is small.

Another Example

Compute $y = \sinh(x) = \frac{1}{2}(e^x - e^{-x})$.

- Naively computing y at an x near 0 may result in a (meaningless) 0.
- Instead use Taylor's expansion

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \dots$$

to obtain

$$\sinh(x) = x + \frac{x^3}{6} + \dots .$$

- If x is near 0, can use $x + \frac{x^3}{6}$, or even just x, for an effective approximation to $\sinh(x)$.

So, a good library function would compute $\sinh(x)$ by the regular formula (using exponentials) for $|x|$ not very small, and by taking a term or two of the Taylor expansion for $|x|$ very small.

Another Example

Compute $y = \sqrt{x+1} - \sqrt{x}$ for $x = 1 \cdot 10^5$ in $\mathbb{F}$ with $t = 5$.

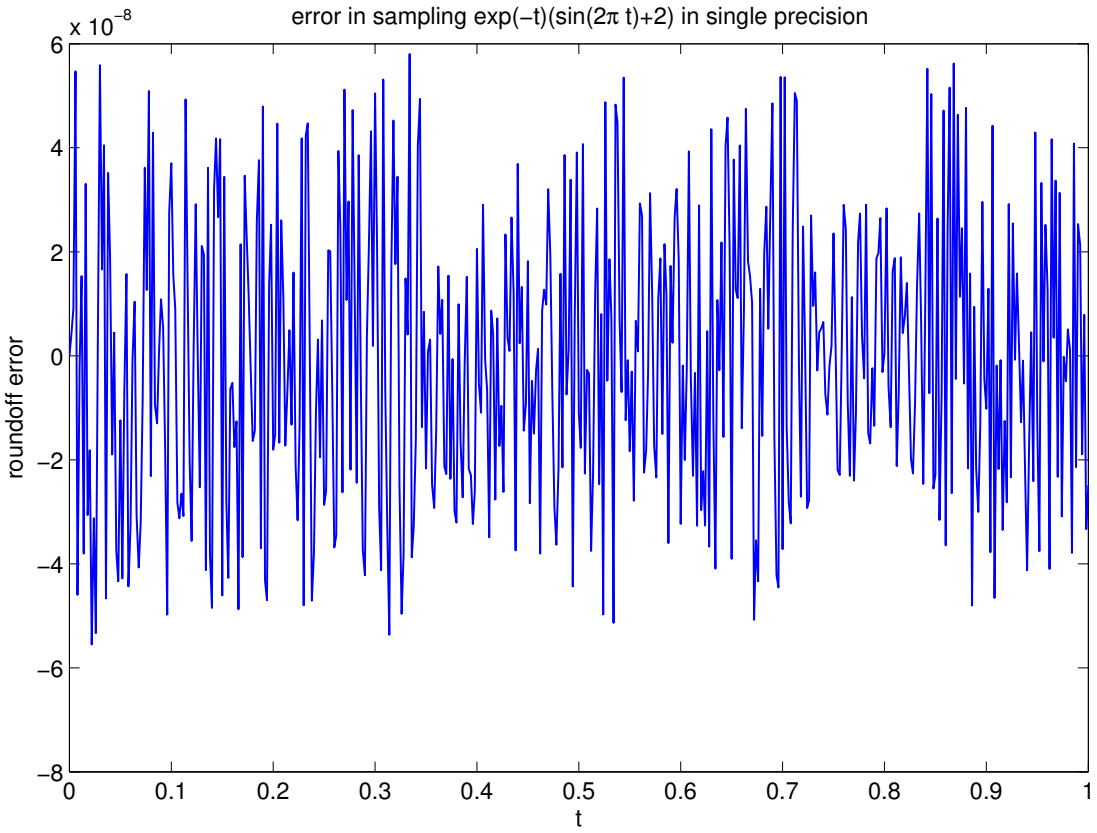
However, $x + 1 = 1.00001 \cdot 10^5 \notin \mathbb{F}$ and $\mathsf{fl}(x+1) = 1 \cdot 10^5 = x$. This results in $\mathbb{F}$ in $x + 1 = x$ and hence $y = 0$.

Instead use the identity

$$\sqrt{x+1} - \sqrt{x} = \frac{(\sqrt{x+1} - \sqrt{x})(\sqrt{x+1} + \sqrt{x})}{(\sqrt{x+1} + \sqrt{x})} = \frac{1}{\sqrt{x+1} + \sqrt{x}}.$$

The Rough Appearance of Roundoff Errors

Run program Example2_2Figure2_2.m



Note how the sign of the floating point representation error at nearby arguments t fluctuates as if randomly: as a function of t it is a "non-smooth" error.

Avoiding Overflow

Suppose $a \gg b$ and we wish to compute $c = \sqrt{a^2 + b^2}$.

Then it may be better to rescale and compute

$$c = a\sqrt{1 + (b/a)^2}.$$

This principle is actually used in the computation of the ℓ_2 norm of a vector:

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Floating Point Arithmetic: Summary

- The order in which operations are performed matters.
  - Algebraically equivalent formulas may have very different roundoff errors, depending on their arguments.
  - For a single operation, roundoff error is usually small (around ϵ_{mach}), but poor formulas can make it **much** bigger in just a few operations (such as through cancellation).
- Algorithms using floating point will always be approximate, even if there is no discretization error.
- Algorithm design must keep error accumulation in mind.
  - Computers can run **a lot** of operations very fast.
  - Computational errors can compound and not just add up.

Outline

Reflection

On a piece of paper, answer the following question:

> *What was the muddiest point in the first course topic on "Numerical Algorithms & Errors"?*

Put the paper into the "Suggestion Box" or just on the table.

Group Exercise 3: Bisection Method

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. Outcome

2. Applications

3. Problem Setting

Outline

1. Outcome
2. Applications
3. Problem Setting

Today's Goal

Practices and further insight on the first method we learned for solving basic, simply stated nonlinear problems.

Outline

Applications of Nonlinear Equations in One Variable

Projectile motion: An artillery officer wants to fire his cannon on an enemy brigade camped d meters away. At what angle θ to the horizontal should he aim the cannon in order to strike the target?

$$f(\theta) = \frac{2v_0^2 \sin(\theta)\cos(\theta)}{g} - d = 0$$

Applications of Nonlinear Equations in One Variable

Financial mathematics: Problems involving loans, interest rates, and savings.

Suppose you borrow $\$100,000$ to buy a house (not in Vancouver!). Each month you are willing to pay p dollars, and the loan is to be paid off in 30 years (360 months). You might ask at what interest rate you can afford the loan.

$$g(r) = 100,000\, r - \frac{(1+r)^{360}}{(1+r)^{360} - 1} = 0$$

Outline

Setting

Consider the scalar, nonlinear equation $f(x) = 0$, where f is continuous on the interval $[a, b]$. We denote a solution of the equation by x^*.

Assume that f changes sign on $[a, b]$, i.e., $f(a) \cdot f(b) < 0$.

We apply the bisection method starting with the interval $[a, b]$. Let x_k be an approximation to x^* obtained after k steps of the bisection method. x_k is the midpoint of the kth subinterval trapping the root.

Lecture Notes 09: Nonlinear Equations in One Variable

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

Fixed Point Iteration

This is an intuitively appealing approach which often leads to simple algorithms for complicated problems.

1. Write given problem
$$f(x) = 0$$
as
$$g(x) = x,$$
so that $f(x^*) = 0$ iff $g(x^*) = x^*$.

2. Iterate:
$$x_{k+1} = g(x_k), \quad k = 0, 1, \ldots,$$
starting with guess x_0.

It's all in the choice of the function g.

Choosing the Function g

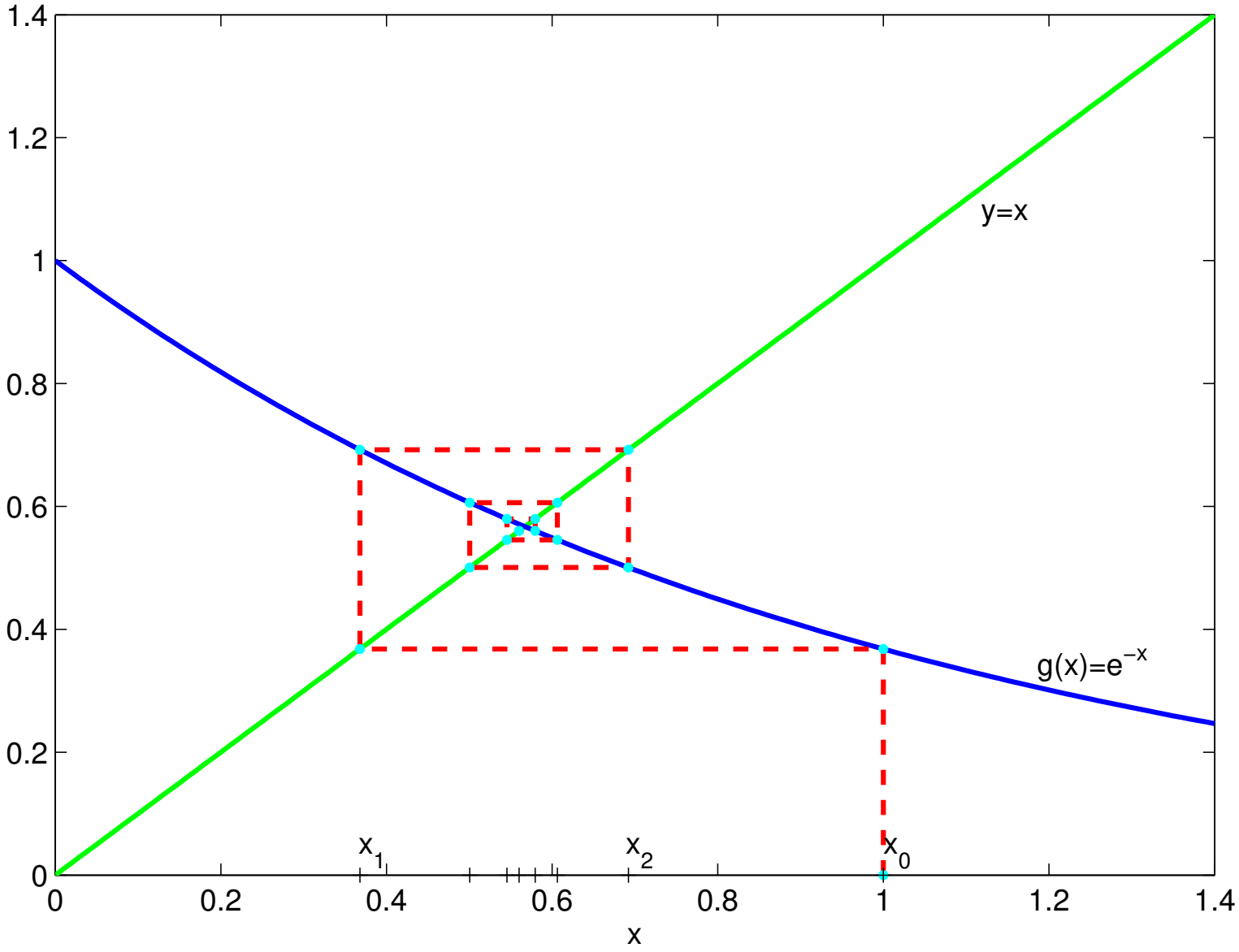
- Note: there are many possible choices g for the given f: this is a family of methods.
- Examples:

$$\begin{aligned} g(x) &= x - f(x), \\ g(x) &= x + 2f(x), \\ g(x) &= x - f(x)/f'(x) \quad (\text{assuming f}' \text{ exists and f}'(\text{x}) \neq 0). \end{aligned}$$

  The first two choices are simple, the last one has potential to yield fast convergence (we'll see later).

- Want resulting method to
  - be simple;
  - converge; and
  - do it rapidly.

Graphical Illustration, $x = e^{-x}$, Starting from $x_0 = 1$



Fixed Point Theorem

Existence:
If $g \in C[a,b]$, $g(a) \geq a$ *and* $g(b) \leq b$, *then there is a fixed point* x^* *in the interval* $[a,b]$.

Uniqueness:
If, in addition, the derivative g' *exists and there is a constant* $\rho < 1$ *such that the derivative satisfies*

$$|g'(x)| \leq \rho \quad \forall\, x \in (a,b),$$

then the fixed point x^* *is unique in this interval.*

Convergence of Fixed Point Iteration

- Assuming $\rho < 1$ as for the fixed point theorem, obtain

$$|x_{k+1} - x^*| = |g(x_k) - g(x^*)| \leq \rho |x_k - x^*|.$$

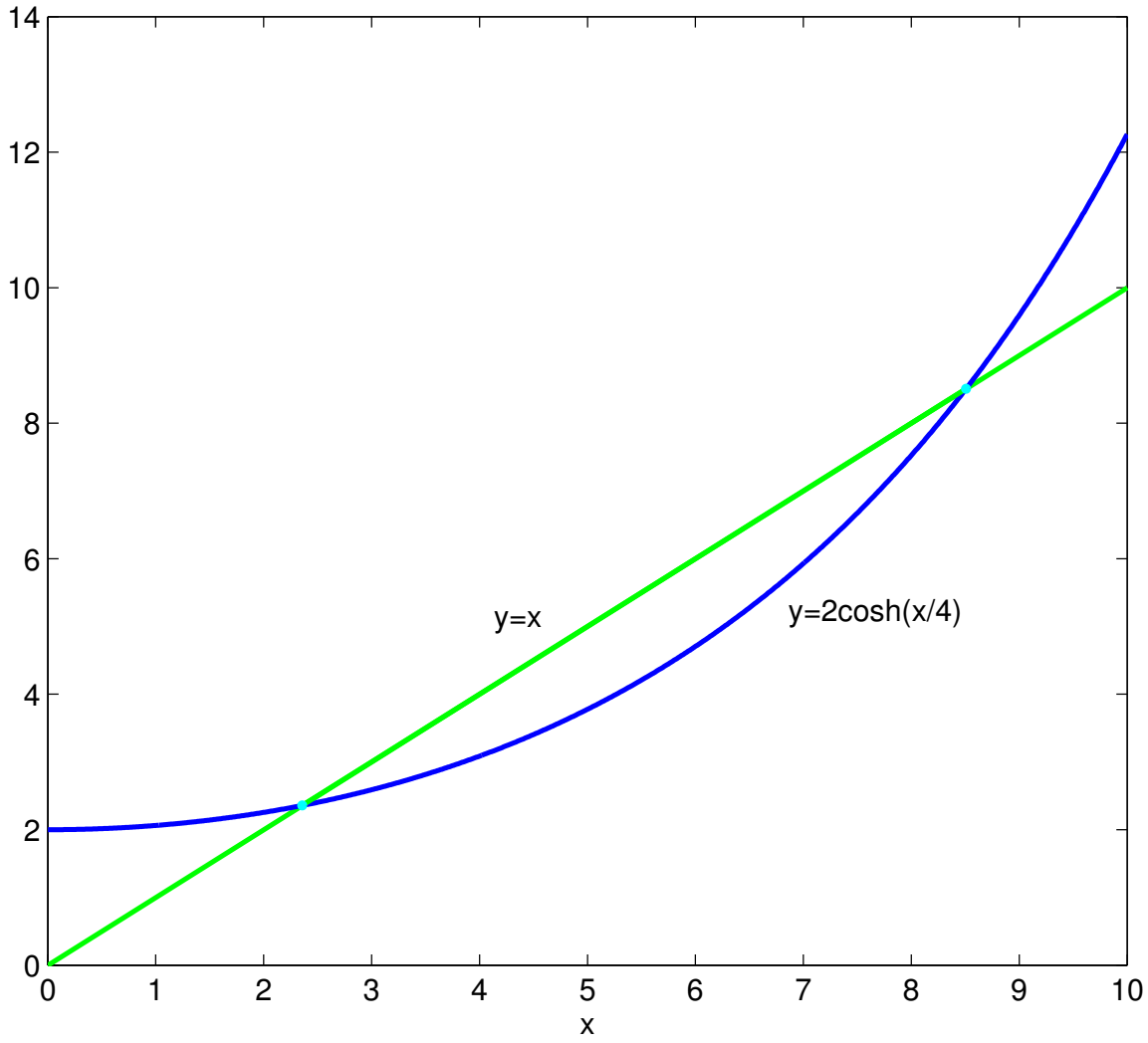
- This is a **contraction** by factor ρ.
- So

$$|x_{k+1} - x^*| \leq \rho |x_k - x^*| \leq \rho^2 |x_{k-1} - x^*| \leq \ldots \leq \rho^{k+1} |x_0 - x^*| \to 0.$$

- The smaller ρ the faster convergence is.

Example: Cosh with Two Roots

$$f(x) = g(x) - x, \quad g(x) = 2\cosh(x/4)$$



Fixed Point Iteration with g

For absolute error tolerance 10^{-8}:

- Starting at $x_0 = 2$ converge to x_1^* in 16 iterations.
- Starting at $x_0 = 4$ converge to x_1^* in 18 iterations.
- Starting at $x_0 = 8$ converge to x_1^* (even though x_2^* is closer to x_0).
- Starting at $x_0 = 10$ obtain **overflow** in 3 iterations.

Note: bisection yields both roots in 27 iterations.

Rate of Convergence

- Suppose we want $|x_k - x^*| \approx 0.1|x_0 - x^*|$.
- Since $|x_k - x^*| \leq \rho^k |x_0 - x^*|$, want

$$\rho^k \approx 0.1,$$

  i.e., $k \log_{10} \rho \approx -1$.

- Define the **rate of convergence** as

$$rate = -\log_{10} \rho.$$

- Then it takes about $k = \lceil 1/rate \rceil$ iterations to reduce the error by more than an order of magnitude.

Return to Cosh Example

- Bisection: $rate = -\log_{10} 0.5 \approx 0.3 \quad \Rightarrow \quad k = 4$.
- For the root x_1^* of fixed point example, $\rho \approx 0.31$ so

$$rate = -\log_{10} 0.31 \approx 0.5 \quad \Rightarrow \quad k = 2.$$

- For the root x_2^* of fixed point example, $\rho > 1$ so

$$rate = -\log_{10}(\rho) < 0 \quad \Rightarrow \quad \text{no convergence.}$$

Lecture Notes 10: Nonlinear Equations in One Variable

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

Chapter 2: Nonlinear Equations in One Variable

- Bisection method
- Fixed point iteration
- Newton's method and variants
- Minimizing a function in one variable

Outline

Outline

Newton's Method

This fundamentally important method is everything that bisection is not, and vice versa:

- Not so simple
- Not very safe or robust (local convergence)
- Requires more than continuity on f
- Fast
- Automatically generalizes to systems

Derivation

- By Taylor series ($h = x - x_k$)

$$f(x) = f(x_k + h) = f(x_k) + f'(x_k)(x - x_k) + f''(\xi(x))(x - x_k)^2/2.$$

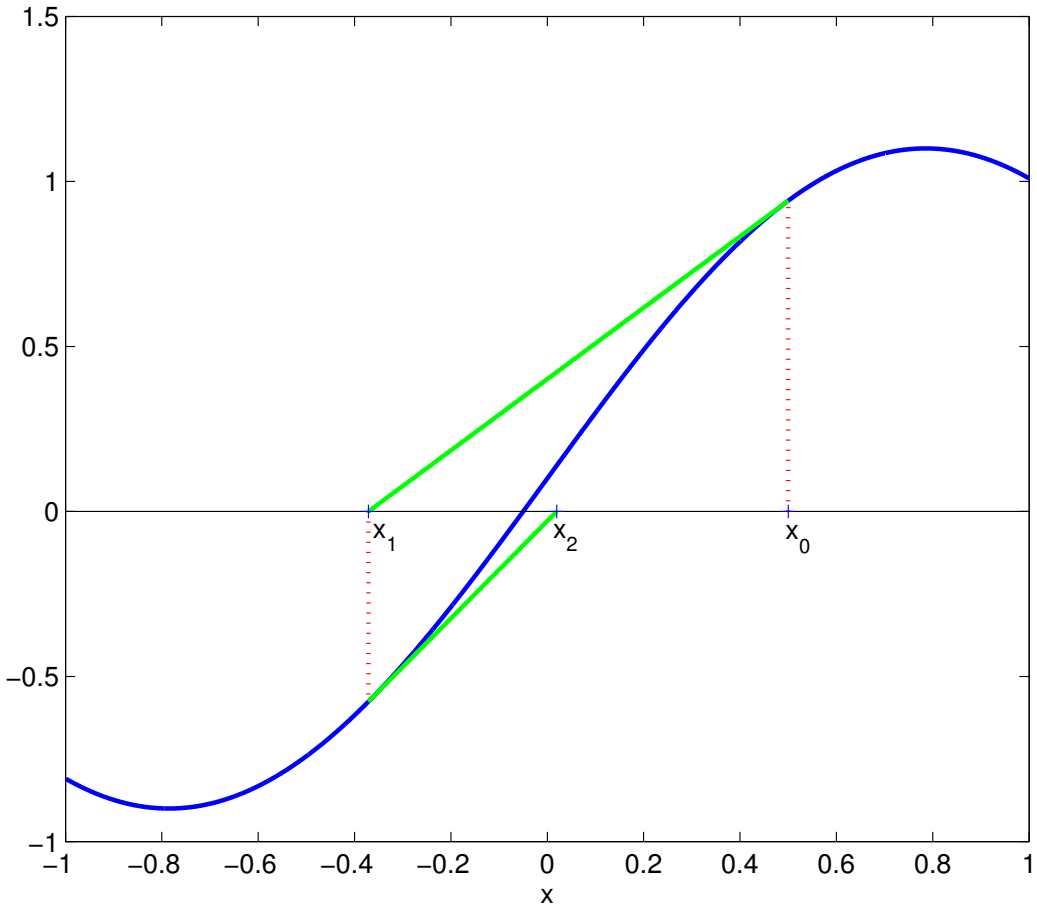
- So, for $x = x^*$

$$0 = f(x_k) + f'(x_k)(x^* - x_k) + \mathcal{O}\left((x^* - x_k)^2\right).$$

- The method is obtained by neglecting nonlinear term, defining $0 = f(x_k) + f'(x_k)(x_{k+1} - x_k)$, which gives the iteration step

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots.$$

A Geometric Interpretation



Next iterate is x-intercept of the tangent line to f at current iterate.

Example: Cosh with Two Roots

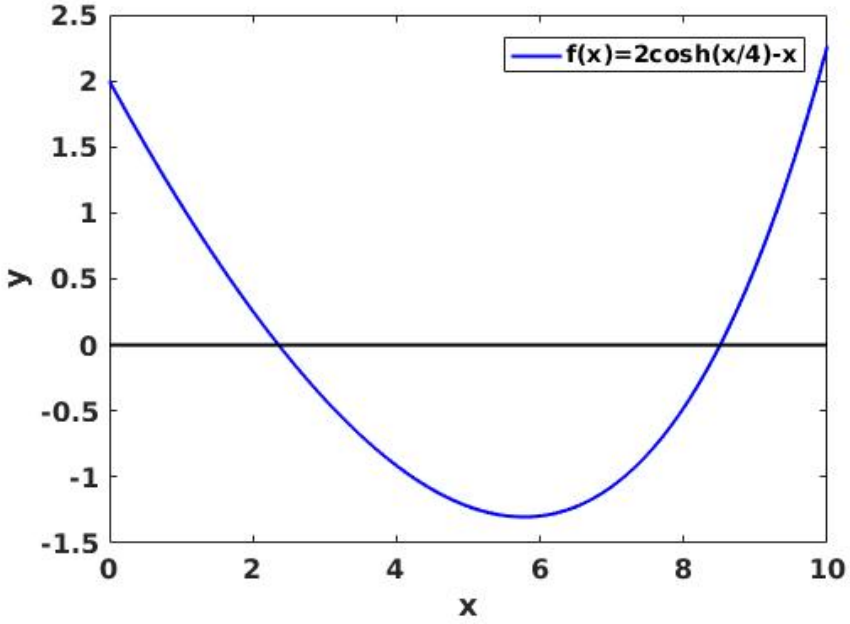
- The function

$$f(x) = 2\cosh(x/4) - x$$

has two solutions in the interval $[2, 10]$.

- Newton's iteration is

$$x_{k+1} = x_k - \frac{2\cosh(x_k/4) - x_k}{0.5\sinh(x_k/4) - 1}.$$



For absolute tolerance 1.e-8:

- Starting from $x_0 = 2$ requires 4 iterations to reach x_1^*.
- Starting from $x_0 = 4$ requires 5 iterations to reach x_1^*.
- Starting from $x_0 = 8$ requires 5 iterations to reach x_2^*.
- Starting from $x_0 = 10$ requires 6 iterations to reach x_2^*.

Example (cont.): Cosh with Two Roots

- Tracing the iteration's progress for $x_0 = 8$:

k	0	1	2	3	4	5
$f(x_k)$	-4.76e-1	8.43e-2	1.56e-3	5.65e-7	7.28e-14	1.78e-15

- Note that the number of significant digits essentially doubles at each iteration (until the 5th, when roundoff error takes over).

Speed of Convergence

A given method is said to be

- **linearly convergent** if there is a constant $\rho < 1$ such that

$$|x_{k+1} - x^*| \leq \rho |x_k - x^*| \ ,$$

  for all k sufficiently large;

- **superlinearly convergent** if there is a sequence of constants $\rho_k \to 0$ such that

$$|x_{k+1} - x^*| \leq \rho_k |x_k - x^*|,$$

  for all k sufficiently large.

- **quadratically convergent** if there is a constant M such that

$$|x_{k+1} - x^*| \leq M |x_k - x^*|^2 \ ,$$

  for all k sufficiently large;

Convergence Theorem for Newton's Method

If $f \in C^2[a,b]$ and there is a root x^* in $[a,b]$ such that $f(x^*) = 0$, $f'(x^*) \neq 0$, then there is a number δ such that, starting with x_0 from anywhere in the neighborhood $[x^* - \delta, x^* + \delta]$, Newton's method converges quadratically.

Idea of proof:

- Expand $f(x^*)$ in terms of a Taylor series about x_k;
- divide by $f'(x_k)$, rearrange, and replace $x_k - \frac{f(x)}{f'(x_k)}$ by x_{k+1};
- find the relation between $e_{k+1} = x_{k+1} - x^*$ and $e_k = x_k - x^*$.

Outline

Secant Method

- One potential disadvantage of Newton's method is the need to *know and evaluate* the derivative of f.
- The secant method circumvents the need for explicitly evaluating this derivative.
- Observe that near the root (assuming convergence)

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

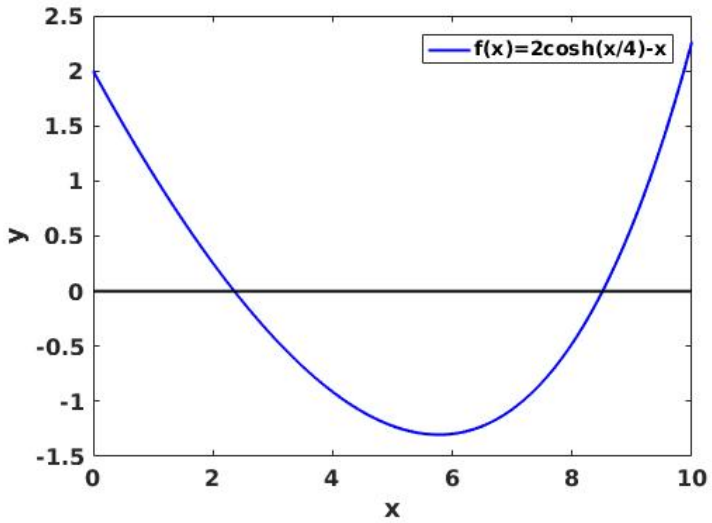
- So, define Secant iteration

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}, \quad k = 1, 2, \ldots.$$

- Note the need for two initial starting iterates x_0 and x_1: a *two-step method*.

Example: Cosh with Two Roots

$$f(x) = 2\cosh(x/4) - x.$$



Same absolute tolerance 1.e-8 and initial iterates as before:

- Starting from $x_0 = 2$ and $x_1 = 4$ requires 7 iterations to reach x_1^*.
- Starting from $x_0 = 10$ and $x_1 = 8$ requires 7 iterations to reach x_2^*.

k	0	1	2	3	4	5	6
$f(x_k)$	2.26	-4.76e-1	-1.64e-1	2.45e-2	-9.93e-4	-5.62e-6	1.30e-9

Observe superlinear convergence: much faster than bisection and simple fixed point iteration, yet not quite as fast as Newton's iteration.

Newton's Method as a Fixed Point Iteration

- If $g'(x^*) \neq 0$ then fixed point iteration converges linearly, as discussed before, as $\rho > 0$.
- Newton's method can be written as a fixed point iteration with

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

  From this we get $g'(x^*) = 0$.
- In such a situation the fixed point iteration may converge faster than linearly: indeed, Newton's method converges quadratically under appropriate conditions.

Lecture Notes 12: Review Linear Algebra

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

Goals of this Chapter

- Provide common background (no numerical algorithms) in linear algebra, necessary for developing numerical algorithms elsewhere.
- Collect several concepts and definitions for easy referencing.
- Ensure that those who have the necessary background can easily skip this chapter.

Chapter 4: Review Linear Algebra

- Basic Concepts: Linear Systems and Eigenvalue Problems
- Norms

- Special Matrix Classes
- Singular Value Decomposition (SVD)
- Examples in Applications

Outline

Outline

Linear Independence and Linear Space

- Consider m real-valued vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$, each of length n, i.e., $\mathbf{v}_i \in \mathbb{R}^n$.
- These vectors are linearly independent if $\sum_{i=1}^{m} \alpha_i \mathbf{v}_i = \mathbf{0}$ necessarily implies that all (real, scalar) coefficients α_i are equal to zero.
- Assuming linear independence, the linear space $V = \text{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \ldots \mathbf{v}_m\}$ consists of all linear combinations $\sum_{i=1}^{m} \alpha_i \mathbf{v}_i$.
- The linear space V has dimension m. It is obviously a subspace of $\mathbb{R}^n$. (Hence, necessarily, $m \leq n$.)

Example

Are

$$\mathbf{v}_1 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}, \qquad \mathbf{v}_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

linear independent?

Consider

$$\mathbf{0} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 = \alpha_1 \begin{pmatrix} 2 \\ 5 \end{pmatrix} + \alpha_2 \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

(I) $2\alpha_1 + \alpha_2 = 0 \quad \Leftrightarrow \quad \alpha_2 = -2\alpha_1$

(II) $5\alpha_1 + \alpha_2 = 5\alpha_1 - 2\alpha_1 = 3\alpha_1 = 0 \quad \Leftrightarrow \quad \alpha_1 = 0$

(I)' $\alpha_2 = 0$

Hence, $\mathbf{v}_1$ and $\mathbf{v}_2$ are linear independent.

Linear System of Equations

- Find $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ which satisfies

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1, \\ a_{21}x_1 + a_{22}x_2 &= b_2, \end{aligned}$$

  or $A\mathbf{x} = \mathbf{b}$ with $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$.

- Unique solution **iff** lines are not parallel.
- Generalize to $n \times n$ systems $A\mathbf{x} = \mathbf{b}$,

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix},$$

  and the elements $a_{ij} \equiv a_{i,j}$ are real numbers.

Linear System of Equations (cont.)

In general, for a square $n \times n$ system with $A \in \mathbb{R}^{n \times n}$ there is a unique solution if one of the following equivalent statements hold:

- A is nonsingular;
- $\det(A) \neq 0$;
- A has linearly independent columns or rows;
- there exists an inverse A^{-1} satisfying $AA^{-1} = I = A^{-1}A$;
- $\text{range}(A) = \mathbb{R}^n$;
- $\text{null}(A) = \{\mathbf{0}\}$.

Example

What is the nullspace of $A = \begin{pmatrix} 2 & 8 \\ 5 & 20 \end{pmatrix}$?

Bring A to triangular form:

$$A = \left.\begin{pmatrix} 2 & 8 \\ 5 & 20 \end{pmatrix}\right| \begin{matrix} \cdot 5 \\ \cdot (-2) \end{matrix} + \quad \rightsquigarrow \quad \tilde{A} = \begin{pmatrix} 2 & 8 \\ 0 & 0 \end{pmatrix}$$

Hence

$$2x_1 + 8x_2 = 0 \Leftrightarrow x_1 = -4x_2$$

and

$$\text{null}(A) = \left\{ \alpha \begin{pmatrix} -4 \\ 1 \end{pmatrix} \middle| \alpha \in \mathbb{R} \right\} .$$

Outline

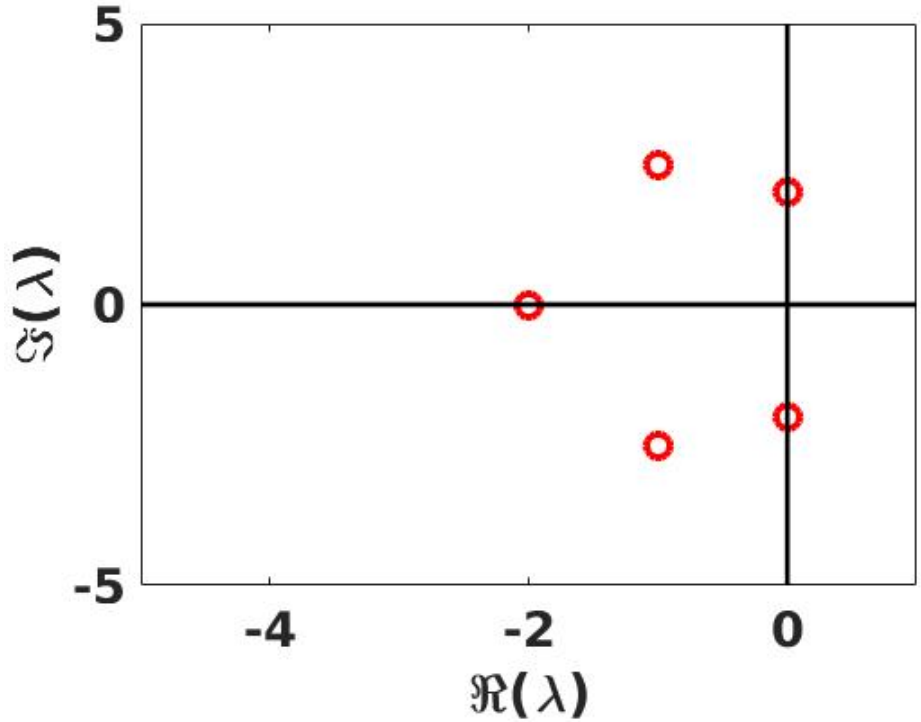
Eigenvalue Problems

Consider $A \in \mathbb{R}^{n \times n}$. A scalar $\lambda \in \mathbb{C}$ and a vector $\mathbf{0} \neq \mathbf{x} \in \mathbb{C}^n$ are an eigenvalue-eigenvector pair (or eigenpair) of A if

$$A\mathbf{x} = \lambda\mathbf{x}.$$

Example: Spectrum of A

$$A = \begin{pmatrix} -0.9880 & 1.8000 & -1.1777 & 0.5662 & -0.1852 \\ -1.9417 & -0.5835 & -0.2830 & -0.6202 & 1.0868 \\ 0.7845 & 1.0422 & -0.5945 & -1.9064 & -0.4689 \\ 0.1628 & 0.9532 & 1.7775 & -0.1745 & 0.5301 \\ -0.7548 & -0.4258 & 0.9639 & -0.1903 & -1.6595 \end{pmatrix}$$



See: spectrum.ipynb.

Eigenvalue Problems (cont.)

- For a **diagonalizable** $A \in \mathbb{R}^{n \times n}$ there are n (generally complex-valued) eigenpairs $(\lambda_j, \mathbf{x}_j)$, with $X = \left[\mathbf{x}_1, \ldots, \mathbf{x}_n\right]$ nonsingular, and $X^{-1}AX$ is a diagonal matrix with the eigenvalues on the main diagonal.

- **Similarity transformation:** Given a nonsingular matrix S, the matrix $S^{-1}AS$ has the same eigenvalues as A. (Exercise: what about the eigenvectors?)

Symmetric Matrices

$A \in \mathbb{R}^{n \times n}$ is symmetric if $A^T = A$.

- **Examples**:
  1. $A = \begin{pmatrix} 4 & 1.01 \\ 1.01 & -2 \end{pmatrix}$
  2. $A = B^T B$ for any real rectangular B.
- **Properties**:
  1. The eigenvalues of A are all real.
  2. The matrix A is always diagonalizable, i.e., it has a full set of n linearly independent eigenvectors.
  3. The eigenvector matrix X can be scaled so that it is real and

$$X^{-1} = X^T.$$

(orthogonal)

Outline

Outline

Vector Norms

A **vector norm** is a function "$\|\cdot\|$" from $\mathbb{R}^n$ to $\mathbb{R}$ that satisfies:

1. $\|\mathbf{x}\| \geq 0; \quad \|\mathbf{x}\| = 0$ iff $\mathbf{x} = \mathbf{0}$,
2. $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\| \quad \forall\ \alpha \in \mathbb{R}$,
3. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|, \quad \forall\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

- This generalizes absolute value or magnitude of a scalar.
- The distance between vectors $\mathbf{x}$ and $\mathbf{y}$ can be measured by $\|\mathbf{x} - \mathbf{y}\|$.

Famous Vector Norms

- ℓ_2-*norm*

$$\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T\mathbf{x}} = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2}$$

- ℓ_∞-*norm*

$$\|\mathbf{x}\|_\infty = \max_{1\leq i\leq n} |x_i|$$

- ℓ_1-*norm*

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

Example

- Problem: Find the distance between

$$\mathbf{x} = \begin{pmatrix} 11 \\ 12 \\ 13 \end{pmatrix} \quad \text{and} \quad \mathbf{y} = \begin{pmatrix} 12 \\ 14 \\ 16 \end{pmatrix}.$$

- Solution: let

$$\mathbf{z} = \mathbf{y} - \mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix},$$

  and find $\|\mathbf{z}\|$.
- Calculate

$$\begin{aligned} \|\mathbf{z}\|_1 &= 1 + 2 + 3 = 6, \\ \|\mathbf{z}\|_2 &= \sqrt{1 + 4 + 9} \approx 3.7417, \\ \|\mathbf{z}\|_\infty &= 3. \end{aligned}$$

Outline

Matrix Norms

Induced matrix norm of $m \times n$ matrix A for a given vector norm:

$$\|A\| = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|} \quad = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|.$$

Then consistency properties hold,

$$\|AB\| \leq \|A\|\|B\|, \quad \|A\mathbf{x}\| \leq \|A\|\|\mathbf{x}\|,$$

in addition to the previously stated three norm properties.

Famous Matrix Norms

- *ℓ_2-norm*

$$\|A\|_2 = \sqrt{\rho(A^T A)} = \sigma_{\max}(A),$$

where ρ is **spectral radius**

$$\rho(B) = \max\{|\lambda|; \ \ \lambda \text{ is an eigenvalue of } B\}.$$

(A relatively painful calculation for such a simple task.)

- *ℓ_∞-norm*

$$\|A\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|.$$

- *ℓ_1-norm*

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|.$$

Lecture Notes 13: Review Linear Algebra

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

Chapter 4: Review Linear Algebra

- Basic Concepts: Linear Systems and Eigenvalue Problems
- Norms

- Special Matrix Classes
- Singular Value Decomposition (SVD)
- Examples in Applications

Outline

Outline

Symmetric Positive Definite Matrices

Extend notion of positive scalar to matrices:

$$A = A^T, \quad \mathbf{x}^T A \mathbf{x} > 0, \quad \forall\, \mathbf{x} \neq \mathbf{0}.$$

A symmetric matrix is positive definite if and only if all its eigenvalues are positive:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n > 0.$$

Useful Facts about Matrices

Consider a real square $n \times n$ matrix $A = (a_{ij})$.

- If $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, then the matrix is nonsingular iff $d = det(A) = a_{11}a_{22} - a_{12}a_{21} \neq 0$. If $d \neq 0$ then

$$A^{-1} = \frac{1}{d} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}.$$

- The matrix is strictly diagonally dominant if for all $i,\ i = 1, 2, \ldots, n$,

$$|a_{ii}| > \sum_{j \neq i} |a_{i,j}|.$$

- Let A be a symmetric, strictly diagonally dominant matrix whose diagonal elements are all positive. Then A is symmetric positive definite.

Outline

Orthogonal and Orthonormal Vectors

- **Orthogonal vectors**: two vectors $\mathbf{u}$ and $\mathbf{v}$ of the same length are orthogonal if

$$\mathbf{u}^T\mathbf{v} = 0.$$

- **Orthonormal vectors:** if *also* $\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1$.

Orthogonal Matrices

- Square matrix Q is **orthogonal** if its columns are pairwise orthonormal, i.e.,

$$Q^T Q = I. \quad \text{Hence also } Q^{-1} = Q^T.$$

- Important properties: for any orthogonal matrix Q and vector $\mathbf{x}$

$$\|Q\mathbf{x}\|_2 = \|\mathbf{x}\|_2. \quad \text{Hence } \|Q\|_2 = \|Q^{-1}\|_2 = 1.$$

- For any symmetric matrix A there is a real orthogonal eigenvector matrix X, so that $X^{-1}AX$ is diagonal.

Outline

1. Special Matrix Classes
2. Singular Value Decomposition (SVD)
3. Examples in Applications

Singular Value Decomposition

Let A be real $m \times n$ (rectangular in general). Then there are orthogonal matrices U $(m \times m)$ and V $(n \times n)$ such that

$$A = U\Sigma V^T,$$

where

$$\Sigma = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}, \quad S = \mathrm{diag}\{\sigma_1, \ldots, \sigma_r\},$$

with the **singular values** satisfying

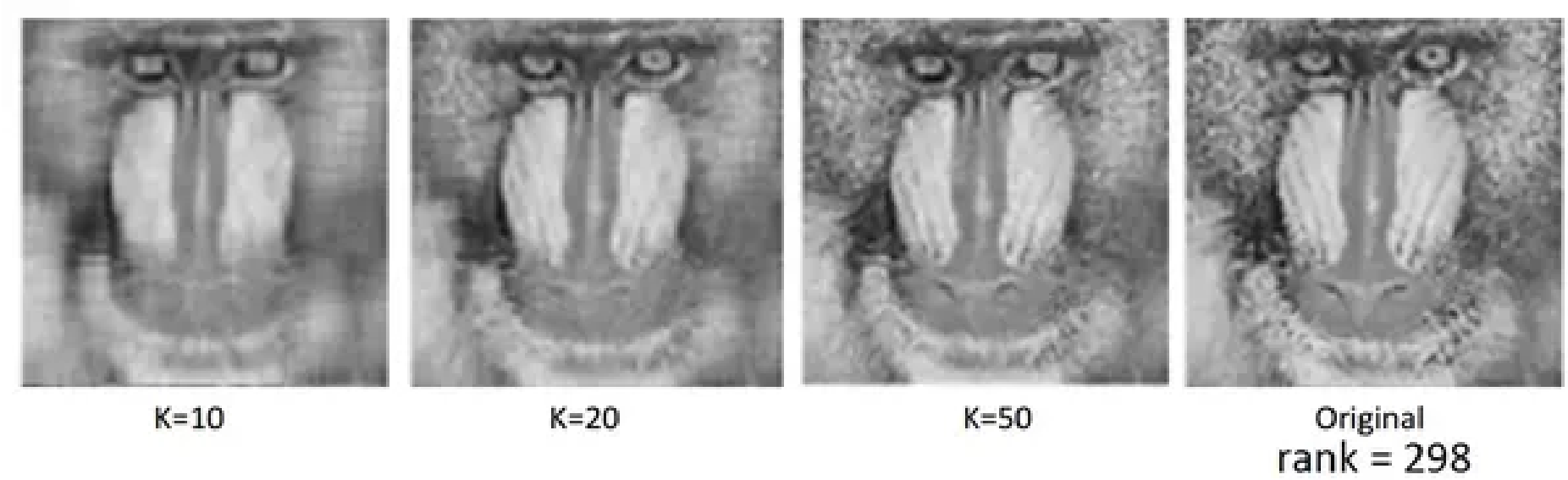
$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0, \quad \sigma_{r+1} = \cdots = \sigma_n = 0.$$

Connection to eigenvalues: $\sigma_i = \sqrt{\lambda_i}$, where λ_i are eigenvalues of $A^T A$.

Outline

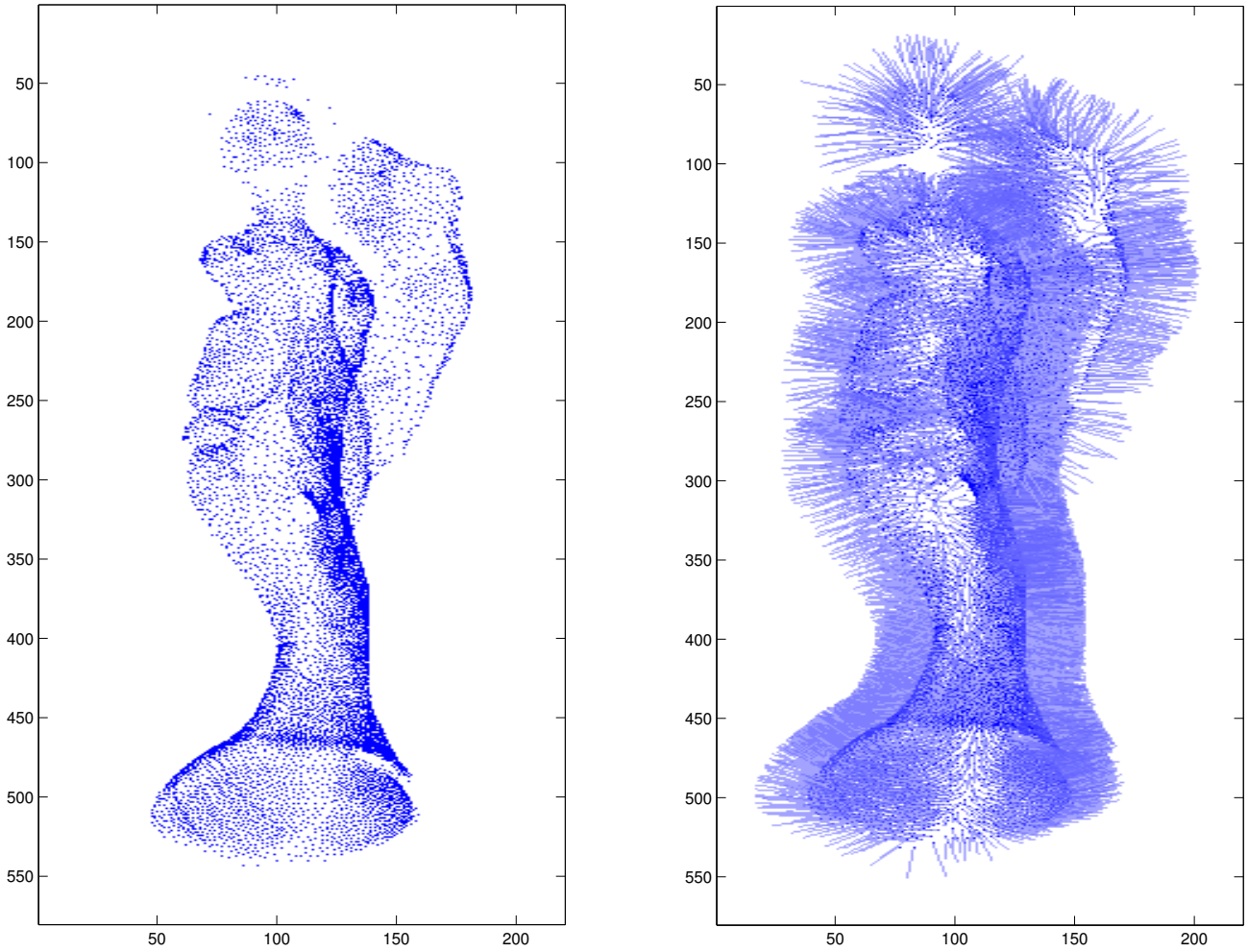
Outline

Low Rank Approximation



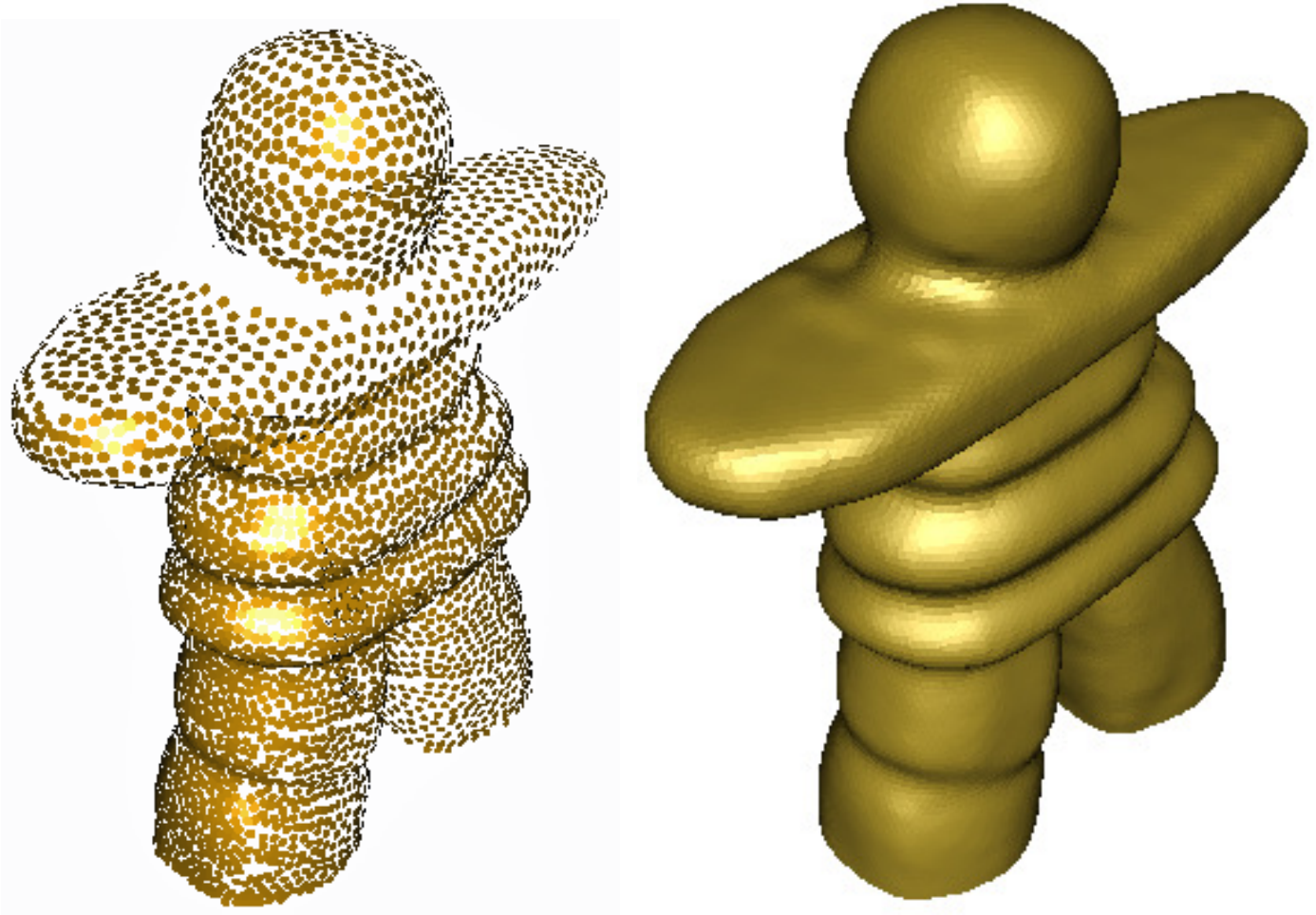
Outline

Instance: Point Cloud



Instance: RBF Interpolation

Left: consolidated point cloud. Right: RBF surface.



Outline

Data Fitting

Given measurements, or observations

$$(t_1, b_1), (t_2, b_2), \ldots, (t_m, b_m) = \{(t_i, b_i)\}_{i=1}^{m},$$

want to fit a function

$$v(t) = \sum_{j=1}^{n} x_j \phi_j(t),$$

For example, polynomial fit:

$$v(t) = x_1 + x_2 t + x_3 t^2 + \ldots + x_n t^{n-1}, \quad \text{so } \phi_j(t) = t^{j-1}.$$

- $\phi_1(t), \phi_2(t), \ldots, \phi_n(t)$ are known linearly independent **basis functions**
- $x_1, \ldots, x_n$ are **coefficients** to be determined s.t. (hopefully)

$$v(t_i) = b_i, \quad i = 1, 2, \ldots, m.$$

Data Fitting (cont.)

Define $a_{ij} = \phi_j(t_i)$. Want $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

Assume that A has full column rank n.

1. If $m = n$ get **interpolation problem**. Use methods of Chapters 5 or 7 to solve
$$A\mathbf{x} = \mathbf{b}.$$

2. If $m > n$ want, e.g., $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2$. Get **least squares data fitting**. Use methods of Chapter 6 to solve
$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2.$$

Outline

Differential Equation

Given $g(t), \;\; 0 \leq t \leq 1$, recover $v(t)$ satisfying $-v'' = g$.

Require two boundary conditions

1. $v(0) = v(1) = 0$, or
2. $v(0) = 0$, $v'(1) = 0$.

Discretize on mesh $t_i = ih, \;\; i = 0, 1, \ldots, N$:

$$-\frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} = g(t_i), \quad i = 1, 2, \ldots, N-1.$$

Note $h = 1/(N+1)$. So, smaller h means larger number of linear equations.

With BC $v(0) = v(1) = 0$, require $v_0 = v_N = 0$.

Linear System for Differential Equation

Need to solve $A\mathbf{v} = \mathbf{g}$, where

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N-2} \\ v_{N-1} \end{pmatrix}, \ \mathbf{g} = \begin{pmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_{N-2}) \\ g(t_{N-1}) \end{pmatrix}, \ A = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}.$$

Thus, A is **tridiagonal**.

Group Exercise 5: Gaussian Elimination and Backward Substitution

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. Outcome
2. New: Bonus Points
3. Reflection

Outline

Today's Goal

Get familiar (again) with the ancient method of Gaussian elimination.

Outline

Bonus Points in Group Exercises

- If you don't get the full number of points, you have the chance to get an additional point if you **fully** completed an optional task.
- The maximum number of points stays the same as given on the sheet.
- Details on the next slide.

Details – Example: Group Exercise 5

- You got less than 3 points in rubric **"Calculations"**.

  If you solved the optional **task 5** completely and correctly, you get an additional point in that rubric.

- You got less than 3 points in rubric **"Justification"**.

  If you solved the optional **task 6** completely and correctly, you get an additional point in that rubric.

Outline

Reflection

On a piece of paper, answer the following question:

What important question do you still have?

Put the paper into the "Suggestion Box" or just on the table.

Lecture Notes 15: Direct Methods for Linear Systems

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

Goals of this Chapter

- Learn practical methods to handle the most common problem in numerical computation.
- Get familiar (again) with the ancient method of Gaussian elimination in its modern form of LU decomposition, and develop pivoting methods for its stable computation.
- Consider LU decomposition in the very important special cases of symmetric positive definite and sparse matrices.
- Study the expected quality of the computed solution, introducing as we go the fundamental concept of a condition number.

Motivation

- Here and in Chapter 6 we consider the problem of finding $\mathbf{x}$ which solves

$$A\mathbf{x} = \mathbf{b},$$

  where A is a given, real, nonsingular, $n \times n$ matrix, and $\mathbf{b}$ is a given, real vector.

- *Such problems are ubiquitous in applications*!

Motivation (cont.)

Two solution approaches:

- *Direct methods*: yield exact solution in absence of roundoff error.
  – Variations of **Gaussian elimination**.
  – Considered in this chapter

- *Iterative methods*: iterate in a similar fashion to what we do for nonlinear problems.
  – Use only when direct methods are ineffective.
  – Considered in Chapter 6

Chapter 4: Direct Methods for Linear Systems

- Gaussian Elimination and Backward Substitution
- LU Decomposition
- Pivoting Strategies
- Efficient Implementation
- Estimating Errors and Condition Number
- Cholesky Decomposition
- Sparse Matrices

Outline

1. Review Gaussian Elimination

2. LU Decomposition

Gaussian Elimination

$$(A|\mathbf{b}) = \left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & b_2 \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} & b_3 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n(n-1)} & a_{nn} & b_n \end{array}\right)$$

$\Downarrow$ eliminate 1st column

$$(A^{(1)}|\mathbf{b}^{(1)}) = \left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & a_{32}^{(1)} & a_{33}^{(1)} & \cdots & a_{3n}^{(1)} & b_3^{(1)} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{n(n-1)}^{(1)} & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right)$$

$\Downarrow$ eliminate 2nd, . . . , (n-1)th column

$$(A^{(n-1)}|\mathbf{b}^{(n-1)}) = \left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{array}\right)$$

Example

$$(A|\mathbf{b}) = \left(\begin{array}{ccc|c} 3 & -2 & 1 & 1 \\ 1 & 1 & 0 & 4 \\ 1 & -1 & 3 & 2 \end{array}\right)$$

$$l_{21} = {}^{a_{21}}/_{a_{11}} = {}^{1}/_{3}$$
$$\mathbf{a}_{2:}^{(1)} = \mathbf{a}_{2:} - l_{21}\,\mathbf{a}_{1:}$$
$$l_{31} = {}^{a_{31}}/_{a_{11}} = {}^{1}/_{3}$$
$$\mathbf{a}_{3:}^{(1)} = \mathbf{a}_{3:} - l_{31}\,\mathbf{a}_{1:}$$

$$\rightsquigarrow \left(A^{(1)}\middle|\mathbf{b}^{(1)}\right) = \left(\begin{array}{ccc|c} 3 & -2 & 1 & 1 \\ 0 & {}^{5}/_{3} & -{}^{1}/_{3} & {}^{11}/_{3} \\ 0 & -{}^{1}/_{3} & {}^{8}/_{3} & {}^{5}/_{3} \end{array}\right)$$

$$l_{32} = {}^{a_{32}^{(1)}}/_{a_{22}^{(1)}} = -{}^{1}/_{5}$$
$$\mathbf{a}_{3:}^{(2)} = \mathbf{a}_{3:}^{(1)} - l_{32}\,\mathbf{a}_{2:}^{(1)}$$

$$\rightsquigarrow \left(A^{(2)}\middle|\mathbf{b}^{(2)}\right) = \left(\begin{array}{ccc|c} 3 & -2 & 1 & 1 \\ 0 & {}^{5}/_{3} & -{}^{1}/_{3} & {}^{11}/_{3} \\ 0 & 0 & {}^{13}/_{5} & {}^{12}/_{5} \end{array}\right) = \left(U\middle|\mathbf{b}^{(2)}\right)$$

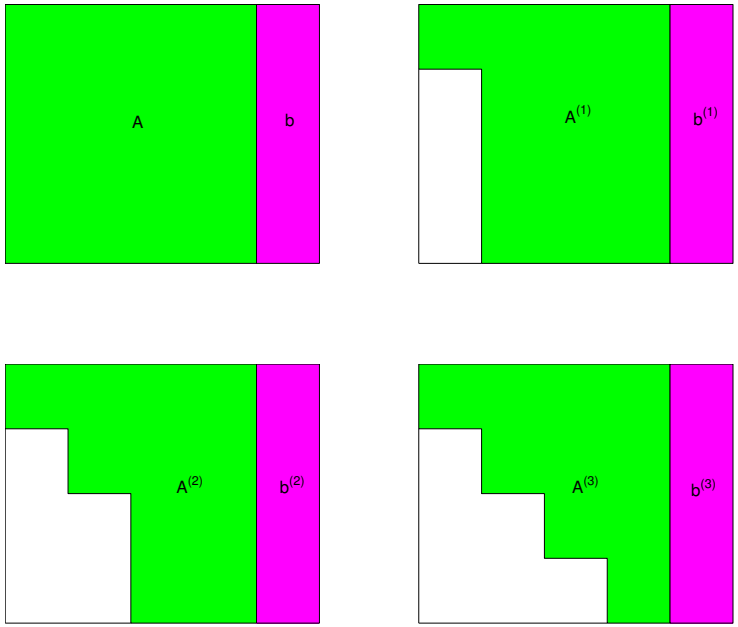
$$A^{(2)} = MA = U$$

Gaussian Elimination (cont.)

- Can multiply a row of $A\mathbf{x} = \mathbf{b}$ by a scalar and add to another row: **elementary transformation**.
- Use this to transform A to upper triangular form:

$$MA\mathbf{x} = M\mathbf{b}, \quad U = MA.$$

- Apply backward substitution to solve $U\mathbf{x} = M\mathbf{b}$.



Outline

Outline

LU Decomposition

- What if we have many right hand side vectors, or we don't know $\mathbf{b}$ right away?
- Note that determining transformation M such that $MA = U$ does not depend on $\mathbf{b}$.

Example (cont.)

$$A = \begin{pmatrix} 3 & -2 & 1 \\ 1 & 1 & 0 \\ 1 & -1 & 3 \end{pmatrix}$$

$$M^{(1)}A = \begin{pmatrix} 1 & 0 & 0 \\ -l_{21} & 1 & 0 \\ -l_{31} & 0 & 1 \end{pmatrix} \begin{pmatrix} 3 & -2 & 1 \\ 1 & 1 & 0 \\ 1 & -1 & 3 \end{pmatrix} = A^{(1)}$$

$$M^{(2)}A^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -l_{32} & 1 \end{pmatrix} \begin{pmatrix} 3 & -2 & 1 \\ 0 & 5/3 & -1/3 \\ 0 & -1/3 & 8/3 \end{pmatrix} = A^{(2)}$$

$$U = A^{(2)} = M^{(2)}A^{(1)} = M^{(2)}M^{(1)}A = MA$$

LU Decomposition (cont.)

$$MA = U$$

$M = M^{(n-1)} \cdots M^{(2)} M^{(1)}$, where $M^{(k)}$ is the transformation of the kth outer loop step. These are elementary lower triangular matrices, e.g.,

$$M^{(2)} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & -l_{32} & \ddots & & \\ & \vdots & & \ddots & \\ & -l_{n2} & & & 1 \end{pmatrix}.$$

Outline

Example (cont.) – Inverse of $M^{(1)}$, $M^{(2)}$

$$M^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ -1/3 & 0 & 1 \end{pmatrix}, \qquad \left[M^{(1)}\right]^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 1/3 & 0 & 1 \end{pmatrix}$$

$$M^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1/5 & 1 \end{pmatrix}, \qquad \left[M^{(2)}\right]^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1/5 & 1 \end{pmatrix}$$

LU Decomposition (cont.)

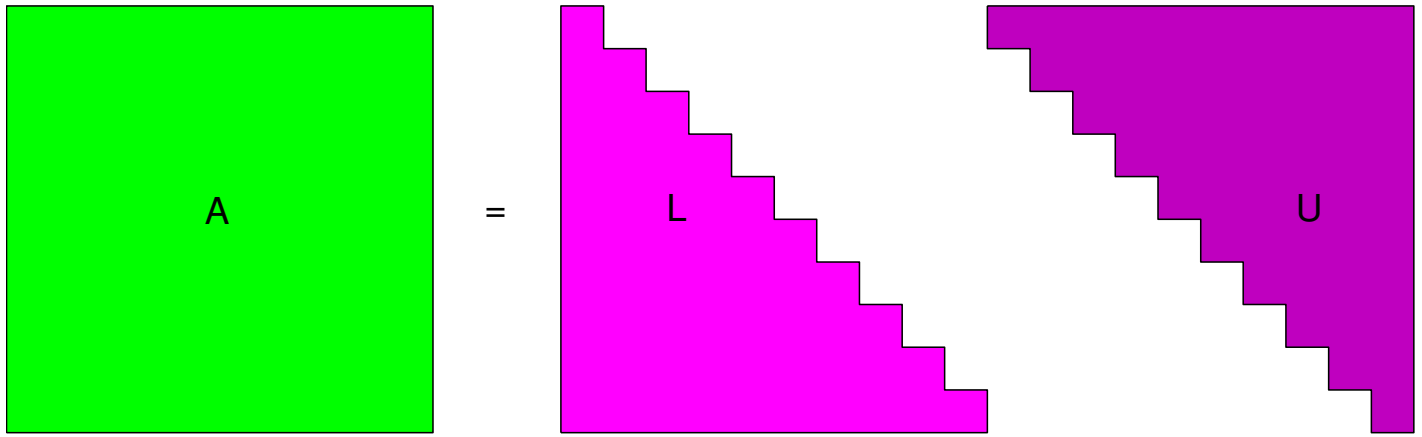
$$MA = U$$

- The matrix $M = M^{(n-1)} \cdots M^{(2)} M^{(1)}$ is unit lower triangular.
- The matrix $L = M^{-1}$ is also unit lower triangular:

$$A = LU, \quad L = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & 1 \end{pmatrix}.$$

Outline

LU Decomposition (cont.)



So, the Gaussian elimination method is equivalent to:

1. decompose $A = LU$. *Now, for a given* $\mathbf{b}$ *we have to solve* $L\big(U\mathbf{x}\big) = \mathbf{b}$:
2. use forward substitution to solve $L\mathbf{y} = \mathbf{b}$;
3. use backward substitution to solve $U\mathbf{x} = \mathbf{y}$.

Examples where the LU Decomposition is Useful

- When we have multiple right-hand sides, form *once* the LU decomposition (which costs $\mathcal{O}(n^3)$ flops); then for each right-hand side only apply forward/backward substitutions (which are computationally cheap at $\mathcal{O}(n^2)$ flops each).

- Can compute A^{-1} by decomposing $A = LU$ once, and then solving $LU\mathbf{x} = \mathbf{e}_k$ for each column $\mathbf{e}_k$ of the unit matrix. These are n right hand sides, so the cost is approximately $\frac{2}{3}n^3 + n \cdot 2n^2 = \frac{8}{3}n^3$ flops.
(However, typically we try to avoid computing the inverse A^{-1}; the need to compute it *explicitly* is rare.)

- Compute determinant of A by

$$\det(A) = \det(L)\det(U) = \Pi_{k=1}^{n} u_{kk}.$$

Another Example (1/3) – Home

$$A = \begin{pmatrix} 1 & -4 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{pmatrix}.$$

Obtain

1. $l_{21} = \frac{1}{1} = 1,\ l_{31} = \frac{3}{1} = 3$, so

$$M^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix}, \quad A^{(1)} = M^{(1)}A = \begin{pmatrix} 1 & -4 & 3 \\ 0 & 5 & -3 \\ 0 & 10 & -8 \end{pmatrix}.$$

2. $l_{32} = \frac{10}{5} = 2$, so

$$M^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix}, \quad A^{(2)} = M^{(2)}A^{(1)} = \begin{pmatrix} 1 & -4 & 3 \\ 0 & 5 & -3 \\ 0 & 0 & -2 \end{pmatrix}.$$

Example (2/3)

- We thus obtain

$$U = A^{(2)} = M^{(2)} A^{(1)} = \begin{pmatrix} 1 & -4 & 3 \\ 0 & 5 & -3 \\ 0 & 0 & -2 \end{pmatrix}.$$

- Note that since $U = M^{(2)} M^{(1)} A$, we have $A = [M^{(1)}]^{-1} [M^{(2)}]^{-1} U$.
- Verify directly that

$$[M^{(1)}]^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}, \quad [M^{(2)}]^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix},$$

and hence that

$$L = [M^{(1)}]^{-1} [M^{(2)}]^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix}.$$

Example (3/3)

- To get the matrix L we therefore collect the multipliers l_{21}, l_{31} and l_{32} into the unit lower triangular matrix

$$L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix}.$$

- Indeed, $A = LU$:

$$\begin{pmatrix} 1 & -4 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & -4 & 3 \\ 0 & 5 & -3 \\ 0 & 0 & -2 \end{pmatrix}.$$

Lecture Notes 17: Direct Methods for Linear Systems

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

Chapter 4: Direct Methods for Linear Systems

- Gaussian Elimination and Backward Substitution
- LU Decomposition
- Pivoting Strategies
- Efficient Implementation

- Estimating Errors and Condition Number

- Cholesky Decomposition
- Sparse Matrices

Outline

Outline

Relative Error in the Solution

- Still consider

$$A\mathbf{x} = \mathbf{b}$$

  but now assess quality of approximate solution obtained somehow.

- Denote exact solution $\mathbf{x}$, computed (or given) approximate solution $\hat{\mathbf{x}}$. Want to estimate

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|}$$

  in some vector norm (∞-, 2-, or 1-norm).

- Can compute the residual $\hat{\mathbf{r}} = \mathbf{b} - A\hat{\mathbf{x}}$ and so also $\frac{\|\hat{\mathbf{r}}\|}{\|\mathbf{b}\|}$.
  Does a small relative residual imply small relative error in solution?

Example

- For the problem

$$A = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 0.8642 \\ 0.1440 \end{pmatrix},$$

  consider the approximate solution

$$\hat{\mathbf{x}} = \begin{pmatrix} 0.9911 \\ -0.4870 \end{pmatrix}.$$

- Then

$$\hat{\mathbf{r}} = \mathbf{b} - A\hat{\mathbf{x}} = \begin{pmatrix} -10^{-8} \\ 10^{-8} \end{pmatrix},$$

  so $\|\hat{\mathbf{r}}\|_\infty = 10^{-8}$.
- However, the exact solution is

$$\mathbf{x} = \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \quad \text{so} \quad \|\mathbf{x} - \hat{\mathbf{x}}\|_\infty = 1.513\,.$$

Outline

Conditioning of Problem

- Since $\hat{\mathbf{r}} = \mathbf{b} - A\hat{\mathbf{x}} = A\mathbf{x} - A\hat{\mathbf{x}} = A(\mathbf{x} - \hat{\mathbf{x}})$, get

$$\mathbf{x} - \hat{\mathbf{x}} = A^{-1}\hat{\mathbf{r}} \quad \text{and hence} \quad \|\mathbf{x} - \hat{\mathbf{x}}\| = \|A^{-1}\hat{\mathbf{r}}\| \leq \|A^{-1}\|\|\hat{\mathbf{r}}\| .$$

- Since $\mathbf{b} = A\mathbf{x}$, get

$$\|\mathbf{b}\| = \|A\mathbf{x}\| \leq \|A\|\|\mathbf{x}\| \quad \text{and hence} \quad \frac{1}{\|\mathbf{x}\|} \leq \frac{\|A\|}{\|\mathbf{b}\|} .$$

- Hence

$$\begin{aligned} \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|} &\leq \|A^{-1}\|\|\hat{\mathbf{r}}\| \frac{\|A\|}{\|\mathbf{b}\|} = \|A^{-1}\|\|A\| \frac{\|\hat{\mathbf{r}}\|}{\|\mathbf{b}\|} = \kappa(A) \frac{\|\hat{\mathbf{r}}\|}{\|\mathbf{b}\|}, \\ \kappa(A) &= \|A\|\|A^{-1}\|. \end{aligned}$$

The scalar $\kappa(A)$ is the **condition number** of A.
For defining $\kappa(A)$ use the induced matrix norm corresponding to the vector norm employed in the above estimates.

Outline

Quality of Solution

- Backward error analysis: associate result of numerical algorithm (GEPP) with the **exact solution** of a **perturbed problem**

$$(A + \delta A)\hat{\mathbf{x}} = \mathbf{b} + \delta\mathbf{b}.$$

- Can rewrite

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \kappa(A) \frac{\|\hat{\mathbf{r}}\|}{\|\mathbf{b}\|}$$

  to

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \kappa(A)\, f(\delta A, \delta\mathbf{b}, \hat{\mathbf{x}})$$

- The job of GEPP is to make δA and $\delta\mathbf{b}$ small.

- Obtain good quality solution (only) if in addition, $\kappa(A)$ is not too large.

Example (cont.)

In our 2×2 example, we saw $\|\mathbf{x} - \hat{\mathbf{x}}\|_\infty \approx 10^8 \, \|\hat{\mathbf{r}}\|_\infty$.

- For the problem

$$A = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix},$$

  we have

$$A^{-1} = 10^8 \begin{pmatrix} 0.1441 & -0.8648 \\ -0.2161 & 1.2969 \end{pmatrix}.$$

- Hence

$$\kappa_\infty(A) = \|A\|_\infty \, \|A^{-1}\|_\infty = 2.1617 \cdot 1.513 \times 10^8 \approx 3.27 \times 10^8$$

  and $\|\mathbf{x} - \hat{\mathbf{x}}\|_\infty \approx \kappa_\infty(A) \, \|\hat{\mathbf{r}}\|_\infty$

Outline

The Condition Number

- Always $\kappa(A) \geq 1$.
- For orthogonal matrices, $\kappa_2(Q) = 1$: ideally conditioned!
- $\kappa(A)$ indicates how close A is to being singular, which $\det(A)$ does not.

Example

Consider

$$A = \begin{pmatrix} 0.01 & & & \\ & 0.01 & & \\ & & \ddots & \\ & & & 0.01 \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

- Remember: A matrix is singular if and only if its determinant is equal to zero.
- $\det(A) = 0.01^n$, and hence close to zero for large n
- **BUT: A is well-conditioned!**, $\kappa(A) = 1$ for all n

The Condition Number (cont.)

- If A is symmetric positive definite with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n > 0$ then

$$\kappa_2(A) = \frac{\lambda_1}{\lambda_n} .$$

- If A is nonsingular with singular values $\sigma_1 \geq \cdots \geq \sigma_n > 0$ then

$$\kappa_2(A) = \frac{\sigma_1}{\sigma_n} .$$

- The exact value of $\kappa(A)$ rarely (if ever) matters in practice: typically, only its order of magnitude is important.

Lecture Notes 18: Direct Methods for Linear Systems

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

Chapter 4: Direct Methods for Linear Systems

- Gaussian Elimination and Backward Substitution
- LU Decomposition
- Pivoting Strategies
- Efficient Implementation
- Estimating Errors and Condition Number

- Cholesky Decomposition

- Sparse Matrices

Outline

Outline

The "Square Root" of a Symmetric Positive Definite Matrix

- Recall that symmetric positive definite is the concept extension of a positive scalar to square real matrices A.
- For a scalar $a > 0$ there is a real square root, i.e., a real scalar g s.t. $g^2 = a$.
- For a symmetric positive definite matrix A, the Cholesky decomposition is written as

$$A = GG^T$$

  where G is a *lower triangular matrix*.

- Obtain Cholesky as a special case of LU decomposition, utilizing both symmetry of A and the assurance that no partial pivoting is needed.

Outline

Example

Consider the symmetric positive definite matrix

$$A = \begin{pmatrix} 1 & \mathbf{w}^T \\ \mathbf{w} & K \end{pmatrix} \in \mathbb{R}^{n \times n} .$$

A single step of Gaussian elimination (introduce zeros in the 1st column) results in

$$A = \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{w} & I \end{pmatrix} \begin{pmatrix} 1 & \mathbf{w}^T \\ \mathbf{0} & K - \mathbf{w}\mathbf{w}^T \end{pmatrix} = \left[M^{(1)} \right]^{-1} A^{(1)} .$$

Gaussian elimination would now continue to introduce zeros in the 2nd column. However, in order to maintain **symmetry**, Cholesky factorization first **introduces zeros in the 1st row** to match the zeros just introduced in the 1st column.

$$A^{(1)} = \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & K - \mathbf{w}\mathbf{w}^T \end{pmatrix} \begin{pmatrix} 1 & \mathbf{w}^T \\ \mathbf{0} & I \end{pmatrix} = \tilde{A}^{(1)} \left[M^{(1)} \right]^{-T} .$$

Example (cont.)

Combining the operations on the previous slide, we find that A has been factored into 3 terms:

$$A = \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{w} & I \end{pmatrix} \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & K - \mathbf{w}\mathbf{w}^T \end{pmatrix} \begin{pmatrix} 1 & \mathbf{w}^T \\ \mathbf{0} & I \end{pmatrix}$$
$$= \left[M^{(1)}\right]^{-1} \tilde{A}^{(1)} \left[M^{(1)}\right]^{-T}.$$

The idea of Cholesky factorization is to continue this process, zeroing one column and one row symmetrically until it is reduced to the identity.

Generalization of the Previous Example

Consider the symmetric positive definite matrix

$$A = \begin{pmatrix} a_{11} & \mathbf{w}^T \\ \mathbf{w} & K \end{pmatrix} \in \mathbb{R}^{n \times n} .$$

Note that $a_{11} > 0$. The generalization of the previous slide is

$$\begin{aligned} A &= \begin{pmatrix} \sqrt{a_{11}} & \mathbf{0}^T \\ \frac{1}{\sqrt{a_{11}}}\mathbf{w} & I \end{pmatrix} \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & K - \frac{1}{a_{11}}\mathbf{w}\mathbf{w}^T \end{pmatrix} \begin{pmatrix} \sqrt{a_{11}} & \frac{1}{\sqrt{a_{11}}}\mathbf{w}^T \\ \mathbf{0} & I \end{pmatrix} \\ &= \left[M^{(1)}\right]^{-1} \tilde{A}^{(1)} \left[M^{(1)}\right]^{-T} . \end{aligned}$$

The second step reads

$$A = \left[M^{(1)}\right]^{-1} \left[M^{(2)}\right]^{-1} \tilde{A}^{(2)} \left[M^{(2)}\right]^{-T} \left[M^{(1)}\right]^{-T} ,$$

and so on, until we reach step n

$$\begin{aligned} A &= \left[M^{(1)}\right]^{-1} \cdots \left[M^{(n)}\right]^{-1} I \left[M^{(n)}\right]^{-T} \cdots \left[M^{(1)}\right]^{-T} \\ &= GG^T . \end{aligned}$$

Outline

Cholesky Decomposition

- Since A is symmetric positive definite, we can stably write

$$A = LU.$$

- Factor out the diagonal of U

$$U = \begin{pmatrix} u_{11} & & & \\ & u_{22} & & \\ & & \ddots & \\ & & & u_{nn} \end{pmatrix} \begin{pmatrix} 1 & \frac{u_{12}}{u_{11}} & \cdots & \cdots & \frac{u_{1n}}{u_{11}} \\ & 1 & \frac{u_{23}}{u_{22}} & \cdots & \frac{u_{2n}}{u_{22}} \\ & & \ddots & & \vdots \\ & & & \ddots & \vdots \\ & & & & 1 \end{pmatrix} = D\tilde{U}.$$

- So $A = LD\tilde{U}$, and by symmetry, $A = LDL^T$.

Cholesky Decomposition (cont.)

- By an elementary linear algebra theorem, $u_{kk} > 0$, $k = 1, \ldots, n$, so can define

$$D^{1/2} = diag\{\sqrt{u_{11}}, \ \ldots \ , \sqrt{u_{nn}}\}.$$

- Obtain $A = GG^T$ with $G = LD^{1/2}$.
- Compute directly, using symmetry, in $\frac{1}{3}n^3 + \mathcal{O}(n^2)$ flops.
- In MATLAB, `R = chol(A)` gives $R = G^T$.

Cholesky Algorithm

Given a symmetric positive definite $n \times n$ matrix A, this algorithm overwrites its lower part with its Cholesky factor.

$$
\begin{array}{l}
\text{for } k = 1 : n \\
\quad a_{kk} = \sqrt{a_{kk}} \\
\quad \text{for } i = k+1 : n \\
\quad\quad a_{ik} = \dfrac{a_{ik}}{a_{kk}} \\
\quad \text{end} \\
\quad \text{for } j = k+1 : n \\
\quad\quad \text{for } i = j : n \\
\quad\quad\quad a_{ij} = a_{ij} - a_{ik}a_{jk} \\
\quad\quad \text{end} \\
\quad \text{end} \\
\text{end}
\end{array}
$$

Lecture Notes 20: Linear Least Squares Problems

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

Goals of this Chapter

- Introduce and solve the linear least squares problem, ubiquitous in data fitting applications.
- Introduce algorithms based on orthogonal transformations.
- Evaluate different algorithms and understand what their basic features translate into in terms of a tradeoff between stability and efficiency.
- Introduce SVD use for rank-deficient and highly ill-conditioned problems.

Chapter 5: Linear Least Squares Problems

- Motivation
- Data Fitting

- Normal Equations
- QR Decomposition
- Householder and Gram-Schmidt
- SVD and Truncated SVD (TSVD)

Outline

Linear Least-Squares

- Throughout this chapter we consider the problem

$$\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2,$$

  where A is $m \times n$, with $m > n$.

- So, it is an overdetermined system of equations: we have more rows, for instance corresponding to data measurements, than columns, where $\mathbf{x}$ corresponds to unknown model parameters.

- In general, there is no $\mathbf{x}$ satisfying $A\mathbf{x} = \mathbf{b}$, hence we seek to minimize a norm of the residual $\mathbf{r} = \mathbf{b} - A\mathbf{x}$. The ℓ_2-norm is the most convenient to work with, although it is not suitable for all purposes, and it enjoys rich theory.

- Assume A has linearly independent columns. Then there is a unique solution to this problem, as we'll soon see.

Outline

Application: Data Fitting

Given measurements, or observations

$$(t_1, b_1), (t_2, b_2), \ldots, (t_m, b_m) = \{(t_i, b_i)\}_{i=1}^{m},$$

want to fit a function

$$v(t) = \sum_{j=1}^{n} x_j \phi_j(t),$$

- $\phi_1(t), \phi_2(t), \ldots, \phi_n(t)$ are known linearly independent basis functions
- $x_1, \ldots, x_n$ are coefficients to be determined s.t.

$$v(t_i) \approx b_i, \quad i = 1, 2, \ldots, m.$$

Data Fitting (cont.)

Define $a_{ij} = \phi_j(t_i)$. Want $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1,1} & a_{m-1,2} & \cdots & a_{m-1,n} \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}, \ \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \ \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{m-1} \\ b_m \end{pmatrix}.$$

Data Fitting (cont.)

Define $a_{ij} = \phi_j(t_i)$. Want $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1,1} & a_{m-1,2} & \cdots & a_{m-1,n} \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}, \ \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \ \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{m-1} \\ b_m \end{pmatrix}.$$

1. If $m = n$ get interpolation problem. Solution $\mathbf{x} = A^{-1}\mathbf{b}$.

Data Fitting (cont.)

Define $a_{ij} = \phi_j(t_i)$. Want $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1,1} & a_{m-1,2} & \cdots & a_{m-1,n} \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}, \ \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \ \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{m-1} \\ b_m \end{pmatrix}.$$

1. If $m = n$ get interpolation problem. Solution $\mathbf{x} = A^{-1}\mathbf{b}$.
2. If $m > n$ we can't generally set $\mathbf{r} = \mathbf{b} - A\mathbf{x} = \mathbf{0}$. So relax requirement: we want, e.g., $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2$. Obtain a least-squares data fitting problem.

Data Fitting (cont.)

Define $a_{ij} = \phi_j(t_i)$. Want $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1,1} & a_{m-1,2} & \cdots & a_{m-1,n} \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{m-1} \\ b_m \end{pmatrix}.$$

1. If $m = n$ get interpolation problem. Solution $\mathbf{x} = A^{-1}\mathbf{b}$.
2. If $m > n$ we can't generally set $\mathbf{r} = \mathbf{b} - A\mathbf{x} = \mathbf{0}$. So relax requirement: we want, e.g., $\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_2$. Obtain a least-squares data fitting problem.

Note: even if we can increase n so that A becomes square, there may be reasons not to want this:

1. A smaller n gives fewer parameters to control and may better describe global trend of data.
2. If the data contains noise, don't want to over-fit it.

Example: Linear Regression

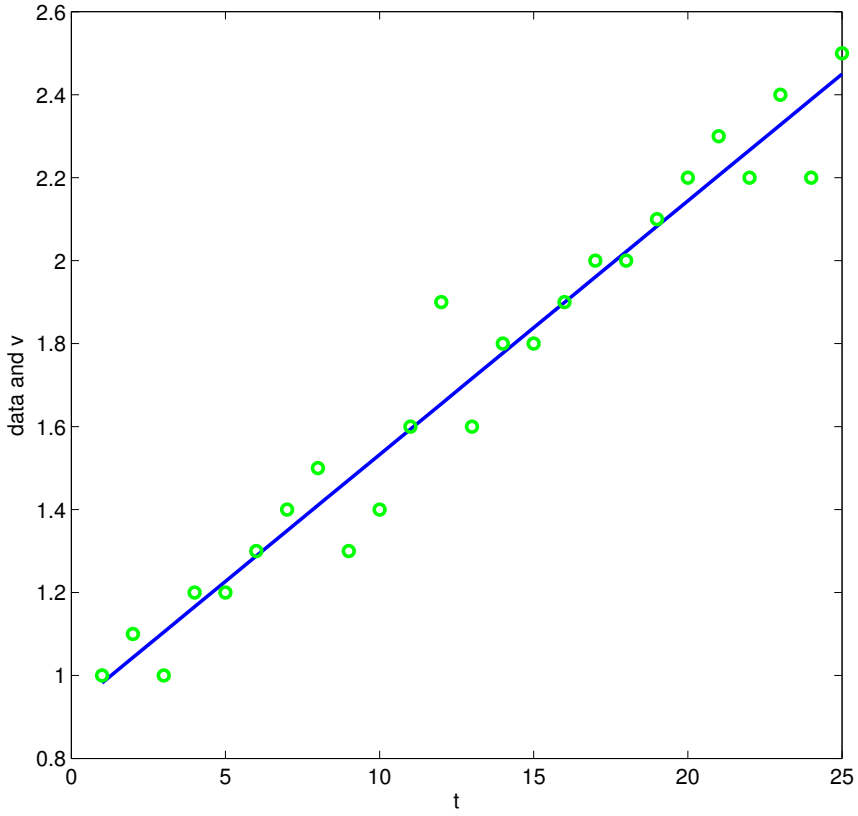


Figure: Linear regression curve (in blue) through green data points. Here $m = 25$ and $n = 2$.

Data Fitting Example

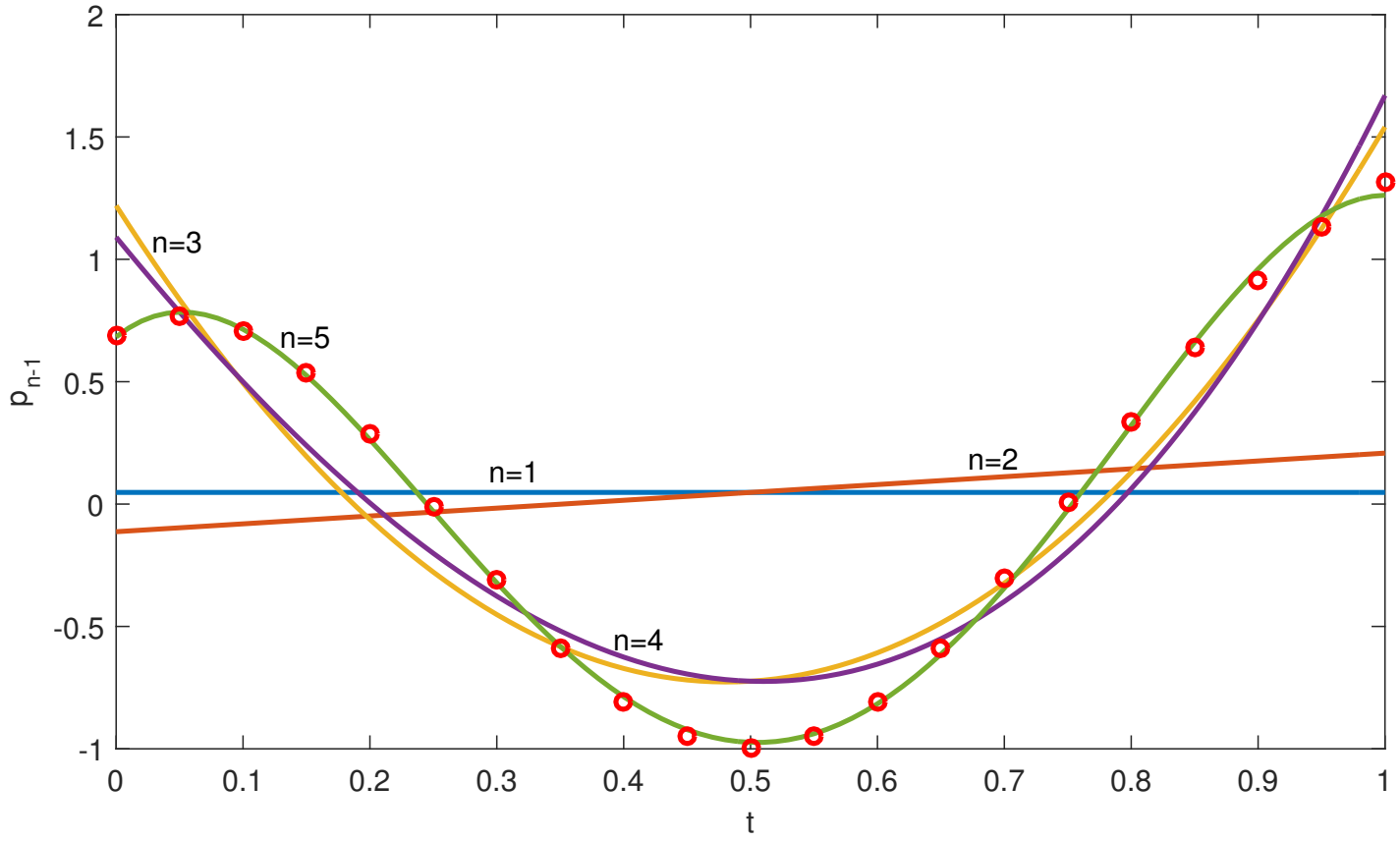


Figure: First 5 best polynomial approximations to $f(t) = \cos(2\pi t) + 10(t - .5)^5$ sampled at $0 : 0.05 : 1$. Data values at red circles.

Other Data Fitting Problems

- When $m > n$, i.e., A "long and skinny", we have an over-determined system.

Other Data Fitting Problems

- When $m > n$, i.e., A "long and skinny", we have an over-determined system.
- Instead of least-squares, may consider

$$\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_1 \quad \text{or} \quad \min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_\infty$$

  for the over-determined problem. Both of these lead to linear programming formulations. The ℓ_1-norm is good against outliers in data.

Other Data Fitting Problems

- When $m > n$, i.e., A "long and skinny", we have an over-determined system.
- Instead of least-squares, may consider

$$\min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_1 \quad \text{or} \quad \min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|_\infty$$

  for the over-determined problem. Both of these lead to linear programming formulations. The ℓ_1-norm is good against outliers in data.
- If $m < n$ we have an under-determined system. Now there are many solutions to $A\mathbf{x} = \mathbf{b}$: want to pick one wisely. For instance,

$$\min_{\mathbf{x}} \{\|\mathbf{x}\|_2 \quad s.t. \quad A\mathbf{x} = \mathbf{b}\}$$

- Alternatively, do it in ℓ_1

$$\min_{\mathbf{x}} \{\|\mathbf{x}\|_1 \quad s.t. \quad A\mathbf{x} = \mathbf{b}\}$$

  Obtain a sparse solution: $x_j = 0$ for at least $m - n$ components. Again, this leads to a linear programming formulation, further discussed in AG, Chapter 9.