Numerical Methods Notes Spring 2021

Contents

Fun Stuff

1. Feynman Method: `https://www.youtube.com/watch?v=FrNqSLPaZLc`
2. Bad math writing: `https://lionacademytutors.com/wp-content/uploads/2016/10/sat-math-section.jpg`
3. Google AI experiments: `https://experiments.withgoogle.com/ai`
4. Babylonian tablet: `https://www.maa.org/press/periodicals/convergence/the-best-known-old-babyl`
5. Parabola in real world: `https://en.wikipedia.org/wiki/Parabola#Parabolas_in_the_physical_world`
6. Parabolic death ray: `https://www.youtube.com/watch?v=TtzRAjW6KO0`
7. Parabolic solar power: `https://www.youtube.com/watch?v=LMWIgwvbrcM`
8. Robots: `https://www.youtube.com/watch?v=mT3vfSQePcs`, riding bike, kicked dog, cheetah, backflip, box hockey stick
9. Cat or dog: `https://www.datasciencecentral.com/profiles/blogs/dogs-vs-cats-image-classificat`
10. History of logarithm: `https://en.wikipedia.org/wiki/History_of_logarithms`
11. Log transformation: `https://en.wikipedia.org/wiki/Data_transformation_(statistics)`

12. Log plot and population: https://www.google.com/publicdata/explore?ds=kf7tgg1uo9ude_&met_y=population&hl=en&dl=en#!ctype=l&strail=false&bcs=d&nselm=h&met_y=population&scale_y=lin&ind_y=false&rdim=country&idim=state:12000:06000:48000&ifdim=country&hl=en_US&dl=en&ind=false

13. Yelp and NLP: https://github.com/skipgram/modern-nlp-in-python/blob/master/executable/Modern_NLP_in_Python.ipynb https://www.yelp.com/dataset/challenge

14. Polynomials and splines: https://www.youtube.com/watch?v=O0kyDKu8K-k, Yoda / matlab, https://www.google.com/search?q=pixar+animation+math+spline&espv=2&source=lnms&tbm=isch&sa=X&ved=0ahUKEwj474fQja7TAhUB3YMKHY8nBGYQ_AUIBigB&biw=1527&bih=873#tbm=isch&q=pixar+animatic mesh+spline, http://graphics.pixar.com/library/

15. Polynomials and pi/taylor series: Matlab/machin https://en.wikipedia.org/wiki/Chronology_of_computation_of_%CF%80 https://en.wikipedia.org/wiki/Approximations_of_%CF%80#Machin-lik formula https://en.wikipedia.org/wiki/William_Shanks

16. Deepfake: face https://www.youtube.com/watch?v=ohmajJTcpNk
    dancing https://www.youtube.com/watch?v=PCBTZh41Ris

17. Pi digit calculations: https://en.wikipedia.org/wiki/Chronology_of_computation_of_%CF%80, poor shanks...https://en.wikipedia.org/wiki/William_Shanks

First day: Introduction to numerical methods

1. What is numerical methods and numerical analysis?
    - Definition: A numerical method is an algorithm to solve problems of continuous mathematics.
    - Definition: Numerical analysis is a theory (theorem / proof) for ensuring quality (convergence, accuracy, efficiency, stability, etc) of a numerical method.
    - The first is our focus.
2. History: The invention of computer and the three math crisis. MAKE BETTER
    - Hippasus:
      https://www.youtube.com/watch?v=P_Na2HUv9ms
    - The ghost step: Weierstrass's function
      https://www.youtube.com/watch?v=pCEFZk9Vihs
      https://www.youtube.com/watch?v=WnaUZrPnZ30
    - Cantor: set theory (Russell, Barber of Seville)
    - Godel:
      https://www.youtube.com/watch?v=la6BK5X2LI8
    - Alan turing: turing machine
      https://www.youtube.com/watch?v=E3keLeMwfHY
3. Why do we need numerical methods? 2 main views.
    - Fields of math
        - Algebra (solve equations)
        - Trig (study functions)
        - Calculus (area, change, infinity)
        - Linear algebra (systems of linear equations)
        - Differential equations (calculus equations of functions)

- Reason 1: Practical (this class)
  - Not solvable: Nonlinear equation, function value ($\sin(1^\circ)$), integration.
  - Data driven problems, can only observe the real world without knowing the true model.
  - Large scale: Large systems of equations (high dimension), regression, AI.
  - The field of scientific computing lives here. Problems from sciences, engineering, economics, business, social science, computer science, AI, etc. are mathematically formulated into these categories. We don't focus on modeling / math formulation from real world. A scientific computing class would do that.
- Reason 2: Theoretical
  - Use an algorithm to explore a theoretical field (pi digit calculations and the study of randomness / essence of numbers)

4. Main topics of this course:
   (a) Chapter 1, Prelims: Computer arithmetic and Taylor series (machine limitations and main theory tool)
   (b) Chapter 2, Linear algebra (life is linear, used for many calculus problems)
   (c) Chapter 3, Root finding methods (used for optimization or tools for bigger problems)
   (d) Chapter 4, Polynomial interpolation (fitting functions to data)
   (e) Chapter 5, Numerical integration (calculus and continuous summation)
   (f) Chapter 6, Spline functions (fitting smooth curves to data, used in engineering)
   (g) Chapter 7, Differential equations (wide range of application)
5. Challenges:
   (a) Want general techniques for all problems, need abstraction.
   (b) Taylor series are key idea of this course.
   (c) Computer makes mistakes. We need to find out why and protect against problems.
      i. Show the examples of sequence illustrating round-off error.
      $$x_0 = \frac{1}{3}, \quad x_{k+1} = \begin{cases} 2x_k, & x_k \in [0, \frac{1}{2}] \\ 2x_k - 1, & x_k \in (\frac{1}{2}], 1 \end{cases}$$
      ii. A computer is really a mathematical model (assumptions are made) and cannot completely capture the real world.
6. Matlab tutorial:
   - `http://www.mathworks.com/support/learn-with-matlab-tutorials.html`
   - Coding suggestions: Section 1.1 of the text.
7. Warnings:
   - This class is by design much more free than past classes. Feedback if you pursue it. Value and skills are yours to gain. You get out what you put in.
   - This course is a heavy workload. Lean on others.
   - Groupwork is essential. Be a good group member.
   - Coding is essential. Be a competent coder. This is the most valuable practical skill gained in this course.
   - Grades are not an issue if you are dedicated: C (valid effort), B (good work), A (excellent work)

Chapter 1 Math preliminaries and floating-point representation

This chapter focuses on two foundation topics:

1. Computer calculations (structure and limitations)
2. Taylor series (same as calculus 2, framework for proving theory of methods)

.1 1.1 Introduction

1. Approach:
   (a) For all the problems listed above, our method will not be to solve as you would in past class (algebra, calculus, etc). Instead we will build iterative algorithms which approach the true solution.
   (b) We already saw the example of machine generated (roundoff) error. Before we find the cause of such error, we need two fundamentals:
      - How to measure accuracy?
      - How to minimize machine operations?
2. Measuring accuracy:
   (a) Precision and significant digits.
      - $\pi \approx 3.141592653589$
      - Def: Significant digits are the digits starting with leftmost nonzero digit ending with the rightmost correct digit.
      - A lot of time approximations are given in scientific notation to equate significant digits to decimal places: $3.1415 = 0.31415 \times 10^1$.
   (b) Absolute and relative errors:
      - Absolute error: $|exact - approximate|$
      - Relative error: $|exact - approximate|/|exact|$.
      - What is the difference? Relative error takes scale into account. How good is the approximation in relation to the exact?
      - Example: Exact=1, approx=1.1
      - Example: Exact=0.1, approx=0.2
   (c) Accuracy and precision:
      - For a sequence of arithmetic operations, can only trust in the result as many significant digits as the least accurate number in the calculation.
      - Example:
        $$(1.2) + (3.45) = 4.65$$
        Result only has two sig digits of accuracy. Reason: Considering rounding, the smaller version of the LHS could be
        $$(1.15) + (3.445) = 4.606$$
        and the larger version as
        $$(1.249) + (3.454) = 4.703$$
        The third digit in the original calculation cannot be trusted.
      - Example: Centennial hall is 70 feet rounded to the nearest tenth. Add a 6 foot tower rounded to the nearest foot. How high is the tower tip? 76ft to one significant digit. Extremes:
        $$65 + 5.5 = 70.5, \quad 74 + 6.4 = 80.4$$
        Strange stuff.

- Lesson: Only round as a final step in calculation.
- Rounding and chopping final results to fit a certain numeric length.

3. Nested multiplication and Horner's algorithm
   (a) Polynomial evaluation $p(2)$ for $p(x) = 2x^3 + 6x^2 - 6x - 18$ requires 6 multiplications and 3 additions. 9 total operations.
   (b) Rewrite as nested multiplication
   $$p(x) = -18 - 6x + 6x^2 + 2x^3 = -18 + x(-6 + x(6 + 2x))$$
   reduces to 3 multiplications and 3 additions.
   (c) In general
   $$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = a_0 + x(a_1 + x(a_2 + \ldots a_n x)))$$
   reduces a n degree polynomial evaluation to n multiplications and n additions. See text for pseudocode.
   (d) Can also use Horner's method to factor linear factors (synthetic division) called deflating a polynomial, as well as compute $p'(x)$ at values. See text.
4. Homework: Ex 1-3, 8, 16, 18; CEx 1-4, 7-10

.2 1.2 Mathematical Preliminaries

1. Idea of power series: Review from Calculus 2
   (a) Represent a function as an infinite series (polynomial)
   $$f(x) = \sum_{n=0}^{\infty} c_n x^n = c_0 + c_1 x + c_2 x^2 + \ldots$$
   (b) Example: Geometric series and Desmos demonstration
   $$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n = 1 + x + x^2 + \cdots, \quad |x| < 1$$
   Important to note that the domain of f may not match the interval of convergence of the series. Hit an asymptote in this case. Center is at zero.
   (c) Example: List you should memorize. Show couple in Desmos.
   $$\begin{aligned} e^x &= \sum_{n=0}^{\infty} \frac{x^n}{n!}, \quad |x| < \infty \\ \sin(x) &= \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!}, \quad |x| < \infty \\ \cos(x) &= \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}, \quad |x| < \infty \\ \ln(1+x) &= \sum_{n=1}^{\infty} \frac{(-1)^{n-1} x^n}{n}, \quad -1 < x \le 1 \\ \arctan(x) &= \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}, \quad -1 \le x \le 1 \end{aligned}$$

(d) Note all intervals are centered at 0. Reason is the power series definition is centered at zero. Can move the center to any $x = a$ as

$$f(x) = \sum_{n=0}^{\infty} c_n (x-a)^n$$

Power series is most accurate near the center as we saw in Desmos.

(e) How to find the interval of convergence? Series convergence tests. Use the ratio test for the interval of convergence and test endpoints with one of the others (integral, alt series, comparison, etc). Recall:

- Example: Find the interval of convergence for $\ln(1+x)$.
- Ratio test for $\sum a_n$: If $\lim a_{n+1}/a_n = L < 1$ the series converges. If $L > 1$ diverges. If $L = 1$ test inconclusive. Can see why endpoints need separate conversation.
- Harmonic series on one side of interval. Alternating harmonic on other.
- Alternating series test for $\sum (-1)^n b_n$: If $b_{n+1} \leq b_n$ and $\lim b_n = 0$, then the series converges. Draw parallel parking intuition.

(f) Given a function, how to get it's power series. 2 ways:

- Use another series. Differentiate sine series to get cosine. Can also integrate. Connect $\ln(1+x)$ and $\arctan(x)$ to geometric series. The radius of convergence is maintained by endpoints must be checked.
- Maclauren / Taylor series gives a formula.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n, \quad f(a) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n$$

Illustrate for e^x series above.

2. Why do we need power/Taylor series? 2 reasons:

(a) Approximation: Truncate a Taylor series to get an approximation of a function.

- Example: $\sin(x) \approx x - \frac{x^3}{3!}$. $\sin(0) = 0$, $\sin(0.1)$, $\sin(\pi/6)$. Better approximation closer to center.
- An important question: How many terms of the Taylor series are needed to ensure a certain accuracy? Taylor's theorem below gives an answer.
- In general,

$$f(x) \approx f(a) + f'(a)(x-a)$$

reformulates as

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

which is the tangent line approximation. Degree 2 Taylor polynomial would be a tangent quadratic?? approximation.

- Often Taylor series are written in terms of h (rather than a) to closer mimic calculus conversation.

$$f(x+h) = \sum_{n=0}^{\infty} \frac{f'(x)}{n!} h^n$$

giving

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

(b) Theory: This is a general framework for ANY function. As long as the power series exists (converges), we can use this framework to prove theorems about numerical methods.

- Example: Euler's method and $y' = 2y, y(0) = 10$ on $[0,1]$. Chop interval into 10, $t_0 = 0, t_1, t_2, \ldots, t_{10} = 1$. Know the true solution at $x = 0$. Approximate $y(t_1) \approx y_1$ by following the tangent line. Result is Euler's method. Beauty is have no idea what y is and can also trade the RHS $2y$ for anything $f(t, y)$.
- Important questions: Does Euler's method converge to the true solution? If yes, how fast? Is EM stable (small changes in initial condition give small changes in solution)? How to improve EM if not satisfied? Taylor's theorem is the key to answering these questions.

3. Taylor's Theorem: If you truncate a Taylor series, how well does that polynomial approximate $f(x)$?

   (a) Theorem:

   $$f(x) = \sum_{k=0}^{n} \frac{f^(k)(a)}{k!}(x-a)^k + E_{n+1}$$

   where error term E_{n+1} has formula

   $$E_{n+1} = \frac{f^{(n+1}(\xi)}{(n+1)!}(x-a)^{n+1}$$

   where ξ lies between x and a.

   (b) What does the error formula say?

   - $n = 0$ case:

     $$f(x) = f(a)(x-a) + E_1 = f(a) + f'(\xi)(x-a)$$

     rearranges as

     $$f'(\xi) = \frac{f(x) - f(a)}{x - a}$$

     for ξ between x and a. This is just the MVT.
   - $n = 1$ case:

     $$f(x) = f(a)(x-a) + f'(x)(x-a) + \frac{f''(\xi)}{2!}(x-a)^2$$

     also incorporates a version of the MVT.
   - MVT in terms of h and big O notation.

     $$f(x+h) = \sum_{k=0}^{n} \frac{f^(k)(x)}{k!} h^k + E_{n+1}$$

     where error term E_{n+1} has formula

     $$E_{n+1} = \frac{f^{(n+1}(\xi)}{(n+1)!} h^{n+1}$$

     where ξ lies between x and $x + h$. Here

     $$|E_{n+1}| \le C h^{n+1}$$

     for some constant C. Often this is written shorthand as

     $$E_{n+1} = \mathcal{O}(h^{n+1})$$

   (c) How to use Taylor's theorem: 3 ways

   - Prove theorems (later).
   - Prove power series convergence to $f(x)$.
   - Find approximation error bounds resulting from truncation.

(d) Example: Show e^x equals it's Maclaurin series.

$$e^x = \sum_{k=0}^{n} \frac{x^n}{n!} + \frac{e^\xi}{(n+1)!}x^{n+1}$$

where ξ is between x and 0. If $\lim E_{n+1} = 0$ then

$$e^x = \lim_{n\to\infty} \sum_{k=0}^{n} \frac{x^k}{k!}$$

and e^x equals it's power series (note infinite series as a limit of a partial sum). For x finite, $\xi < x < B$. Then,

$$\lim_{n\to\infty} |\frac{e^\xi}{(n+1)!}x^{n+1}| \leq \lim_{n\to\infty} \frac{e^B}{(n+1)!}B^{n+1} = 0$$

by the squeeze theorem. Can bound above by $B^2/n \to 0$.

(e) Example: How many terms of power series for $\sin(x)$ are needed to approximate $\sin(1^\circ)$ within 10^{-8}? 10^{-20}? Graph in Desmos to see the task. Make sure to convert degrees to radians. Two ways here: Taylor's theorem and alt series test remainder.

- Since $f(x) = \sin(x)$, we can bound $f^{(n+1)}(\xi) \leq 1$. This is not always doable for any f.
- Then,

$$|E_{n+1}| \leq \left|\frac{1}{(n+1)!}x^{n+1}\right| = \frac{(1^\circ)^{n+1}}{(n+1)!} = \frac{(\pi/180)^{n+1}}{(n+1)!} \leq 10^{-8}$$

- Taking log of both sides, need

$$(n+1)\log(\pi/180) - \log((n+1)!) \leq -8$$

  Put into Desmos with slider for n to find number of terms.

(f) Example: How many terms of power series for $\sin(x)$ are needed to approximate $\sin(1^\circ)$ within 10^{-8}? 10^{-20}? Alt series test remainder approach.

- If alternating series $\sum(-1)^n b_n$ converges with b_n decreasing, then the error after truncating at term n is

$$|E_n| = |S - S_n| \leq b_{n+1}$$

- Check on own to see if this is a tighter approximation to the error.

4. Homework: Exercises 1, 4, 6, 8, 9, 11-14, 16, 35, 38; Computer Exercises 1, 7, 10, 12, 16, 17, 24

.3 1.3 Floating point representation

1. Normalized floating point representation:

(a) Can write a decimal number in normalized floating point representation (scientific notation) as

$$231.825 = 0.231825 \times 10^3$$

(b) In general,

$$x = \pm 0.d_1d_2d_3\cdots \times 10^n = \pm r \times 10^n$$

where $d_1 \neq 0$, $d_k \in \{0, 1, 2, 3, 4, \ldots, 9\}$, and n is an integer. Number $r \in [1/10, 1)$ is called the mantissa and n the exponent.

(c) For a binary number x,

$$x = \pm 0.1b_1b_2\cdots \times 2^m = \pm q \times 2^m$$

for mantissa $q \in [1/2, 1)$. Note the 1 before b_1.

(d) All real numbers in any base can be representative this way. Since computer are finite machines, there is a limit to the size of the mantissa and exponent. As a result only some real numbers can be represented as machine numbers (no irrationals, no inconvenient fractions, no for most real numbers). Also, set of representable machine numbers has strangely spaced gaps.

2. Number systems review:

   (a) Decimal system (base 10):

   $$753.1415 = 7 \times 10^2 + 5 \times 10^1 + 3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2} + 1 \times 10^{-3} + 5 \times 10^{-4}$$

   (b) Binary system (base 2):

   $$(1011.11)_2 = \cdots = 11.75$$

   (c) Likewise for other bases: Octal, hexidecimal.

   (d) Conversion

   - What is 29 in base 2?

     $$29 = 16 + 8 + 4 + 1 = 11101_2$$

     Can also continually divide by 2.

     $$\frac{29}{2} = 14 + \frac{1}{2}, \quad \frac{14}{2} = 7 + \frac{0}{2}, \quad \frac{7}{2} = 3 + \frac{1}{2}, \quad \frac{3}{2} = 1 + \frac{1}{2}, \quad \frac{1}{2} = 0 + \frac{1}{2}$$

     Rephrase in nested (Horner's) form.

     $$29 = 2\times14+1 = 2\times(2\times7+0)+1 = 2\times(2\times[2\times3+1]+0)+1 = 2\times(2\times[2\times(2\times1+1)+1]+0)+1$$

   - Other bases are the same. Try on own. Convert to base 5. $727 = (10402)_5$. Convert $(21401)_5$ to base 10. Check that it is right.

3. Floating-point representation: How does a computer store number x as

   $$x = \pm q \times 2^m$$

   where $q = (0.1b_1b_2 \dots b_k)_2$?

   (a) Toy example: Consider the system which can only store words of the form

   $$\pm 0.1b_1b_2 \times 2^n, \quad n = -1, 0, 1$$

   Representable numbers considering 3 exponents: Both positive and negative versions.

   $$1.00_2 = 1, 2, \frac{1}{2}$$

   $$1.01_2 = \frac{5}{4}, \frac{5}{2}, \frac{5}{8}$$

   $$1.10_2 = \frac{3}{2}, 3, \frac{3}{4}$$

   $$1.11_2 = \frac{7}{4}, \frac{7}{2}, \frac{7}{8}$$

   Draw on number line. Note the non-uniform spacing. Numbers closer to 1 and -1, gap around zero.

   - Gap next to zero is known at the *hole at zero*. Any real number in this gap is rounded to 0 in machine representation. This is known as *underflow*.

- Largest representable number is $\frac{7}{2}$. Any real number larger than this is rounded to ∞ known as *overflow*.
- Machine ϵ gives an upper bound on the relative error due to rounding. This is then the smallest number such that $1+\epsilon \neq 1$. Here this is what?
- Computer needs 5 bits to store all such numbers: One for sign $\pm$, two for exponent e_1, e_2, two for mantissa b_1, b_2.

(b) Standard IEEE floating point representations:

- Single: 32 bits, 1 for sign, 8 for exponent, 23 for mantissa
- Double: 64 bits, 1 for sign, 11 for exponent, 52 for mantissa
- Long double: 80 bits, 1 for sign, 15 for exponent, 64 for mantissa

4. Single precision (32 bit) floating point representation:

(a) Numbers are represented of the form:

$$(-1)^s \times 2^{c-127} \times (1.f)_2$$

where s is the sign bit, f is the binary one-plus mantissa, and c is the biased exponent.

(b) Draw the bits in a 32 bit string in order s, c, f.

- $0 < c < (11111111)_2 < 225$ where endpoints are reserved for special cases (∞, NAN, etc). Then,

$$-126 \leq c - 127 \leq 127$$

are the possible exponents.
- For the mantissa,

$$1 \leq (1.f)_2) \leq (2 - 2^{-23})$$

The largest representable number is then

$$(2 - 2^{-23})2^{127} \approx 2^{128} \approx 3.4 \times 10^{38}$$

and the smallest is

$$2^{-126} \approx 1.2 \times 10^{-38}$$

- Machine ϵ gives an upper bound on the relative error due to rounding. This is then the smallest number such that $1+\epsilon \neq 1$.

$$\epsilon = 2^{-24} \approx 5.96 \times 10^{-8}$$

Then, single precision gives approximately 7 digits of precision.

5. Example: Write 85.125 in single precision.

(a) Convert 85 to binary.

$$\frac{85}{2} = 42 + \frac{1}{2}$$
$$\frac{42}{2} = 21 + \frac{0}{2}$$
$$\frac{21}{2} = 10 + \frac{1}{2}$$
$$\frac{10}{2} = 5 + \frac{0}{2}$$
$$\frac{5}{2} = 2 + \frac{1}{2}$$
$$\frac{2}{2} = 1 + \frac{0}{2}$$
$$\frac{0}{2} = 0 + \frac{1}{2}$$

Combining the remainders we have $85 = (1010101)_2$.

(b) Convert 0.125 to binary.

$$\begin{aligned} 0.125 \times 2 &= 0.25 \\ 0.25 \times 2 &= 0.5 \\ 0.5 \times 2 &= 1.0 (\text{drop the leading } 1) \\ 0.5 \times 0 &= 0 \end{aligned}$$

Combining leading terms we have $0.125 = (0.001)_2$ as expected.

(c) Then,

$$85.125 = (1010101.0001)_2 = (1.0101010001)_2 \times 2^6$$

6. Double precision (64 bits) similar to single precision. Highlights below. Read details on text on own.
   - Largest representable number: $x \leq 1.8 \times 10^{308} \approx 2^{1024}$
   - Smallest representable number: $x \geq 2.2 \times 10^{-308} \approx 2^{-1022}$
   - Machine $\epsilon = 2^{-53} \approx 2.2 \times 10^{-16}$. So we can trust 15 digits in double precision.
7. Caution:
   (a) In short, be mindful and safeguard against roundoff error issues.
   (b) Programming demo in finding limitations of double precision.
   (c) Project 1 explores how to get around the limitations of double precision for π digit calculation.
   (d) Read text on error analysis for floating point representation.
8. Preventing roundoff error.
   (a) For loop coding demo.
   (b) Polynomial plotting coding demo.
9. Homework: Exercises 1,2,5,11,15,17; Computer Exercises 1,2,4,9

.4 1.4 Loss of Significance

Read on own. Simple tricks to avoid machine error.

1. $fl(x)$ is the floating point representation of x.
2. Subtraction:
   - $\sqrt{x^2+1} - 1$ for x near zero. Rationalize numerator instead.
   $$(\sqrt{x^2+1} - 1)\frac{\sqrt{x^2+1}+1}{\sqrt{x^2+1}+1} = \frac{x^2}{\sqrt{x^2+1}+1}$$
   - Taylor series: $x - \sin(x) = \frac{x^3}{3!} - \frac{x^5}{5!} + \ldots$ via sine's Taylor series expansion.
   - Formula: $\cos^2(x) - \sin^2(x) = \cos(2x)$ for x near $\frac{\pi}{4}$.
3. Periodicity: $\sin(x) = \sin(x + 2\pi n)$ for huge / tiny x.
4. Homework: Computer exercises 1, 3, 4, 9

Chapter 2 Linear Systems

Linear algebra is central to many applications, especially data driven problems. Here we focus on how to carefully solve large systems of the form

$$A\vec{x} = \vec{b}.$$

.1 2.1 Naive Gaussian Elimination

1. Linear algebra review:
   (a) Example: Solve the linear system $A\vec{x} = \vec{b}$ where
   $$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 15 \\ 25 \end{bmatrix}$$
   (b) Linear algebra ideas:
   - Augmented matrix
   - Elementary row operations (performing these on an augmented matrix does not change solution space)
     - Add a row to another to replace tha trow
     - Multiply row by a constant
     - Swap two rows
     - Row equivalent: Matrices A and B are row equivalent to each other if A can be transformed into B via only elementary row operations.
   - Pivot entries.
   - Echelon form and reduced row echelon form.
   - Forward elimination and backwards substitution.
   - Determinant and test for invertibility.
   - Solution set (delete a column or row of augmented matrix and discuss multiple / no solutions).

   (c) Existence and Uniqueness Theorem: For system $A\vec{x} = \vec{b}$,
   - Existence: If echelon form of the system is consistent, that is there is no row of the form
   $$\left[\begin{array}{cccc|c} 0 & 0 & \dots & 0 & b \end{array}\right], b \neq 0,$$
   then at least one solution $\vec{x}$ exists.
   - Uniqueness: In general for $A_{m\times n}$, there is a unique solution $\vec{x}$ if the coefficient matrix (in the augmented matrix) has n total pivots. For the $n \times n$ case there are many equivalent statements to ensure uniqueness.
     - $\det(A) \neq 0$
     - A^{-1} exists
     - Columns are linearly independent
     - Rows are linearly independent
     - Range of $T(\vec{x}) = A\vec{x}$ is $\mathbb{R}^n$.
     - $A\vec{x} = \vec{0}$ has only the trivial solution.
     - $\text{null}(A) = \left\{\vec{0}\right\}$
     - $\text{col}(A) = \text{row}(A) = \mathbb{R}^n$
     - RREF of A is I
     - A can be written as a product of elementary matrices (used above, will use later).
     - `https://en.wikipedia.org/wiki/Invertible_matrix`
     - `https://en.wikipedia.org/wiki/Fundamental_theorem_of_algebra`

2. Numerical method: Gaussian elimination and backwards substitution. Show class and ask which is which. Here A is already an augmented matrix.

(a)
```
for i = 1 : m
    if (all the row below are zero)
        break;
    else
        for j = i+1 : m
            row j = row j - row i * A(j,i) / A(i,i);
        end
    end
end
```
(b)
```
for i = n : -1 : 1
    x(i) = A(i,n+1)/A(i,i);
    for j = i -1: -1 : 1
        A(j,n+1) = A(j,n+1) - A(j,i) * x(i);
    end
end
```
(c) Notes:
   - It directly mimics hand computation
   - Efficiency: very slow, many operations needed
   - Accuracy: Can be major problems
     – Dividing by zero (easy to avoid)
     – Accumulation of roundoff error from all the divisions.

3. Notation for measuring accuracy:
   - $\vec{e} = \tilde{}\ \vec{-}\boldsymbol{x}$ is the error vector
   - $\vec{r} = A\vec{\tilde{x}} - \vec{b}$ is the residual vector
   - ℓ^2 norm is standard:
$$\|x\| = \left(\sum x_i^2\right)^{1/2}$$
4. Homework: Exercises: 1, 3, 5, Computer Exercises: 1, 3, 8, 10

.2 2.2 Gaussian Elimination with Scaled Partial Pivoting

1. Accuracy issues with naive Gaussian elimination:
   (a) Roundoff error: If matrix coefficients differ largely in magnitude, information can be lost.
   (b) Instability: Matrices can be "almost" singular (barely invertible). In this case small roundouff can be amplified.
2. Roundoff error: Solving equivalent systems gives different rounding solutions.
   (a) Equivalent systems:
$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$
   (b) Which version is better?
   (c) First version has echelon form
$$\begin{bmatrix} \varepsilon & 1 \\ 0 & 1 - \frac{1}{\varepsilon} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 - \frac{1}{\varepsilon} \end{bmatrix}$$

gives rounding for small ϵ as

$$1 - \frac{1}{\epsilon} \approx -\frac{1}{\epsilon}, \quad 2 - \frac{1}{\epsilon} \approx -\frac{1}{\epsilon},$$

and

$$x_2 = \frac{-1/\epsilon}{-1/\epsilon} = 1.$$

Then,

$$\epsilon x_1 = 1 - x_2 = 1 - 1 = 0$$

and

$$x_1 = 0$$

The second equation of the original system is obviously wrong.

(d) Second version has echelon form

$$\begin{bmatrix} 1 & 1 \\ 0 & 1-\varepsilon \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1-2\epsilon \end{bmatrix}$$

gives rounding for small ϵ as

$$1 - \epsilon \approx 1, \quad 1 - 2\epsilon \approx 1$$

and

$$x_2 = \frac{1}{1} = 1$$

Then,

$$x_1 = 2 - x_2 = 1$$

which is a better solution.

(e) Why is the second strategy better? Choosing the largest pivot possible in each column makes division avoids loss of significance with subtraction.

(f) Choosing the pivot entry as the largest in that column is known as GE with partial pivoting.

(g) Going one step further, choosing the pivot as the largest in column OR row is GE with full (or complete) pivoting.

(h) Partial pivoting is mostly good enough to prevent issue, though it is susceptible to scaling.

$$\begin{bmatrix} 2 & \frac{2}{\epsilon} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{2}{\epsilon} \\ 2 \end{bmatrix}$$

would not be partial pivoted, and issue above endures. Instead the pivot is selected by choosing the largest scaled pivot. Rescaling by the largest coefficient in each row:

$$\frac{2}{2/\epsilon} = \epsilon, \quad 1$$

then would ask for pivoting. Scaled partial pivoting is the common practice in software.

(i) Partial / full / scaled pivoting saves from roundoff issues, though the sacrifice is efficiency.

(j) Example of scaled partial pivoting: Random 3×3 matrix with various positive and negative coefficients. Note, actual pivoting of rows is usually skipped.

3. Instability and matrix condition numbers: Here loss of significance from subtraction is not the issue. Instead small roundoff error is amplified by the matrix in use.

   (a) Famous example: Hilbert matrix. Despite GE with scaled partial pivoting, issues still occur, mostly due to the matrix properties, not roundoff error. Coding demo.

$$H_{ij} = \frac{1}{i+j}$$

(b) Why does this happen? The condition number of H is huge.

$$1 \le \text{cond}(A) < \infty$$

Closer to 1 implies stable. Very large implies instable (nearly nonsingular).

(c) Idea of condition number: Given A, let

$$A\vec{x} = \vec{b} \quad \text{and} \quad A\vec{x_*} = \vec{b_*}$$

for $\|\vec{b} - \vec{b}_*\|$ (residual) small. For well-posedness, we hope the inputs $\vec{x}$ and $\vec{x}_*$ are close. Check.

$$\begin{aligned} A(\vec{x} - \vec{x}_*) &= \vec{b} - \vec{b}_* \\ \vec{x} - \vec{x}_* &= A^{-1}(\vec{b} - \vec{b}_*) \\ \|\vec{x} - \vec{x}_*\| &\le \|A^{-1}\| \cdot \|\vec{b} - \vec{b}_*\| \\ \frac{\|\vec{x} - \vec{x}_*\|}{\|\vec{x}\|} &\le \|A^{-1}\| \cdot \frac{\|\vec{b} - \vec{b}_*\|}{\|\vec{b}\|} \cdot \frac{\|\vec{b}\|}{\|\vec{x}\|} \end{aligned}$$

Note,

$$\frac{\|\vec{b}\|}{\|\vec{x}\|} = \frac{\|A\vec{x}\|}{\|\vec{x}\|} \le \|A\| \frac{\|\vec{x}\|}{\|\vec{x}\|} = \|A\|$$

Finally,

$$\frac{\|\vec{x} - \vec{x}_*\|}{\|\vec{x}\|} \le \|A\| \|A^{-1}\| \cdot \frac{\|\vec{b} - \vec{b}_*\|}{\|\vec{b}\|}$$

and $\text{cond}(A) = \|A\| \|A^{-1}\|$. Note we are looking at relative error here. Also, note $\|A\| = \max \|A\vec{x}\|$ for unit vector $\vec{x}$.

(d) Gaussian elimination with pivoting summary

- Pivoting solve the roundoff error issue
- Pivoting makes Gaussian elimination even slower
- Pivoting for every step is too much
- Pivoting only when $A(i, i)$ is sufficiently small (or zero)
- Matrix condition number is usually checked automatically prior to solving. Warning messages occur.
- In practice, high performance software attempt these in order:
  - Check if matrix A is special form (tridiagonal, triangular)
  - Attempt Cholesky decomposition (efficient)
  - Do LU decomposition (GE reformulated as solving two triangular systems).

4. Homework: Exercises 3, 8, 23, Computer exercises 1-4, 14, 15

.3 2.3 Tridiagonal and Banded Systems

For special structure matrices, GE can be made much more efficient / stable. See text.

Chapter 3: Solving nonlinear equations

With many numerical method, we will see the following structure.

1. Math problem (algebra, calculus, linear algebra, DEs)
2. Method design (based on math idea)

3. Implementation (efficiency, roundoff error)
4. Numerical analysis (theorems on efficiency, error analysis, stability)
5. Experiments (verify analysis, refine design to improve)

.1 3.1 Bisection method

1. Solving even basic looking equations isn't easy.
   - $3x - 2 = 0$
   - $x^2 + 3x - 2 = 0$
   - $x^3 + 3x - 2 = 0$
   - $e^x \sin(x^3) = \arctan(\sqrt{x})$
2. Bisection method idea:
   (a) Math problem: Solve any equation $f(x) = 0$ known as a root finding problem. Note any equation to be solved can be reorganized in this way.
   (b) Method design: Intermediate Value Theorem of calculus: If
   - $f(x)$ is continuous on $[a, b]$ and
   - $f(a) \cdot f(b) < 0$

   then $f(c) = 0$ for some c in (a, b). Draw picture to illustrate.
   (c) Algorithm:
   i. Pick a starting interval $[a, b]$ with $f(a) \cdot f(b) < 0$
   ii. Middle point $c = \frac{a+b}{2}$
   iii. Depending on the value of $f(c)$ pick one of the two subintervals.
   iv. Repeat this procedure until it converges.
   (d) How to know when to stop? We don't know the true root. One way

$$|x_n - x_{n-1}| < TOL$$

   for some error tolerance TOL.
3. Pseudocode:

```
function bisection(f,a,b,tol)

while (|a-b| > tol)
    c = (a+b)/2;
    if f(c) * f(a) < 0
        b = c;
    else
        a = c;
    end
x = c;
return x;
```

4. Numerical analysis:
   (a) Well-posedness: f need be continuous on $[a, b]$ and have a root there. Is there only one?
   (b) The starting points a, b matter. Could miss some zeros.

(c) Accuracy: Does it converge to the real root?

(d) Efficiency: How fast does it converge?

5. Convergence analysis:

   - Function $f(x)$ with root r on interval $[a, b]$
   - n: number of iterations
   - x_n: numerical solution obtained at the n_{th} iteration
   - e_n: error after nth iteration
   $$e_n = x_n - r$$
   - Error analysis: Bound the error. That is, find function $g(n)$ such that
   $$|e_n| < g(n)$$

6. How to bound $|e_n|$? Can handle directly since halving the interval at each step. Draw a number line with $[a, b]$ and root r between. Add $x_1, x_2, \ldots$ then note:
$$|r - x_1| < |a - x_1| = \frac{b-a}{2}, \quad |r - x_2| < \frac{b-a}{4}, \quad \ldots \quad , |r - x_n| < \frac{b-a}{2^n}$$

   Theorem .1 (Convergence analysis of bisection method)**.**
$$|e_n| = |r - x_n| < \frac{|a-b|}{2^n}$$

   Note,

   - It is always convergent assuming the function is continuous with a zero on $[a, b]$
   - Result is called an error estimate. Gives the rate of convergence.
   - This is just an upper bound. May converge faster. It is curriuous that the upper bound is independent of the function all together.
   - How to use this? What if we want $|e_n| < TOL$?
   $$\frac{b-a}{2^n} < TOL \quad \Rightarrow \quad n > \log_2\left(\frac{b-a}{TOL}\right)$$

7. Examples:

   (a) How many operations are needed *at most* for finding the root r of $f(x) = x^3 + 4x^2 - 10$ on $[1, 2]$ within $TOL = 10^{-4}$? How do we know there even is a root? IVT!
   $$n > \log_2\left(\frac{2-1}{10^{-4}}\right) \approx 13.28$$
   So $n = 14$ should do it.
   ```
   bisection(@(x)x^3+4*x^2-10, 1, 2, 1e-4)
   ```

   (b) How many operations are needed so that we are within machine precision?
   ```
   bisection(@(x)x^3+4*x^2-10, 1, 2, eps)
   ```

8. Rate of convergence:

(a) Bisection: Note that since $|e_n| < \dfrac{b-a}{2^n}$, we don't quite have that

$$|e_n| \leq \frac{1}{2}|e_{n-1}|^1$$

We say a method converges linearly if for some constant c,

$$|e_n| \leq c|e_{n-1}|^1$$

For bisection we get convergence and a useful error bound, but we cannot say linear convergence.

(b) In general, we have a convergence rate p if $\lim\limits_{n\to\infty} x_n = r$ and

$$|e_n| \leq c|e_{n-1}|^p$$

We hope to be able to prove this in theory, but we can also measure in practice. Rewriting, we have

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^p} = C$$

Then, for n large enough,

$$|e_{n+1}| \approx C|e_n|^r \quad \Rightarrow \quad \frac{|e_{n+1}|}{|e_n|} \approx \left(\frac{|e_n|}{|e_{n-1}|}\right)^p \quad \Rightarrow \quad \log\left(\frac{|e_{n+1}|}{|e_n|}\right) \approx p\log\left(\frac{|e_n|}{|e_{n-1}|}\right)$$

$$\Rightarrow \quad p \approx \frac{\log\left(\frac{|e_{n+1}|}{|e_n|}\right)}{\log\left(\frac{|e_n|}{|e_{n-1}|}\right)}$$

Define

$$p_n = \frac{\log\left(\frac{|e_{n+1}|}{|e_n|}\right)}{\log\left(\frac{|e_n|}{|e_{n-1}|}\right)}$$

to use in the homework. Another approach is to plot the following log transform and pay attention to the slope of the line which gives the rate of convergence.

$$|e_{n+1}| \approx C|e_n|^r \quad \Rightarrow \quad \log|e_{n+1}| = r\log|e_n| + \log(C) \quad \Rightarrow \quad y = rx + b$$

9. Note: Two extensions of bisection given in the text. See HW.

   (a) False position: Using the zero of the secant line thru a and b rather than the midpoint. ROC improves.

   (b) Modified false position: Prevents repeat endpoints in false position by modifying the secant line.

10. Homework: Exercises 1, 8, 9, 10, Computer exercises 1-3, 6, 8, 21

.2 3.2 Newton's method

1. Idea: Follow the tangent line to find the zero.

   (a) Show Wikipedia animation and ask them to find the iteration formula: `en.wikipedia.org/wiki/Newton%27s_method`

   (b) The linear approximation of $f(x)$ at x_0 is

   $$f(x) \approx l(x) = f(x_0) + f'(x_0)(x - x_0)$$

   This is just the first order Taylor polynomial. Later Taylor's theorem will give a way to analyze performance of Newton's method via this approximation.

(c) Example: The linear approximation of x^2 at $x_0 = 1$

$$l(x) = 2(x-1) + 1 = 2x - 1$$

Draw the picture to illustrate.

2. Newton's method:

   (a) Approximate solving $f(x) = 0$ with solving $l(x) = f(x_0) + f'(x_0)(x - x_0) = 0$ giving the first approximation as

   $$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

   (b) Algorithm

      i. Pick an x_0

      ii. Repeat until convergence

   $$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

   (c) Pseudocode:

```
function newton(f, f', x0, tol)

x = x0;
x0 = x + 2*tol;
while(|x-x0| > tol)
    x0 = x;
    x = x0 - f(x0)/f'(x0);
end

return x;
```

3. Convergence analysis

   (a) Why does it work? Tangent line idea makes sense, but let's check the cases. Draw a parabola with two zeros for example.

   $$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

	$f(x) > 0$	$f(x) < 0$
$f'(x) > 0$	$-$ (dec x_n)	$+$ (inc x_n)
$f'(x) < 0$	$+$ (inc x_n)	$-$ (dec x_n)

   As long as we are close to the zero, we will move in the direction we need. Note if not close enough will diverge or find a different zero.

   (b) How well does it work? Excel demo comparing to Bisection.

   - How to compare two numerical methods?
   - Error table, error graph, log error graph, rate of convergence approximation, theorem results.
   - Comparison is not straightforward. Pros and cons of each.

   (c) Careful analysis: Let r be the root of $f(x) = 0$. Our goal is to connect e_{n+1} to e_n as $|e_{n+1}| < C|e_n|^p$. Then,

   $$e_{n+1} = x_{n+1} - r = x_n - \frac{f(x_n)}{f'(x_n)} - r = e_n - \frac{f(x_n)}{f'(x_n)}$$

   Taking the Taylor expansion of $f(x)$ at x_n,

   $$f(x) = f(x_n) + f'(x_n)(x - x_n) + \frac{f''(\xi)}{2}(x - x_n)^2, \quad x_n < \xi < x$$

Plug in $x = r$

$$0 = f(r) = f(x_n) - f'(x_n)e_n + \frac{f''(\xi)}{2}e_n^2, \quad x_n < \xi < r$$

giving

$$f(x_n) = f'(x_n)e_n - \frac{f''(\xi)}{2}e_n^2$$

Thus

$$e_{n+1} = e_n - \frac{f(x_n)}{f'(x_n)} = \frac{f''(\xi)}{2f'(x_n)}e_n^2$$

(d)

Theorem .2 (Error estimate of Newtons' method)**.** *Newton's method converges if $C(h)$ is bounded and*

$$|e_{n+1}| \leq \frac{1}{2}\frac{\max\limits_{|x-x_n|\leq h}|f''(x)|}{\min\limits_{|x-x_n|\leq h}|f'(x)|}e_n^2 \leq C(h)e_n^2$$

where h is the radius of a small interval containing r.

So we have quadratic rate of convergence.

4. When does Newton's method fail? Coding example.
   - $f(x)$ is not continuous differentiable
   - $f'(x_0) = 0$
   - Cycles
   - Runaway to infinity
   - Initial guess x_0 is crucial.
   - Fail images: `https://www.researchgate.net/profile/Paulo_Flores3/publication/299604992/figure/fig26/AS:613932318343200@1523384279523/Examples-in-which-the-Newton-Raphson-met png`
5. Newton method compared with bisection method
   (a) Cons:
   - Requires $f'(x)$
   - It occasionally fails
   - Initial value must be chosen carefully
     - Need a separated algorithm to do this.

   (b) Pros:
   - Converges faster
   - Can be generalized to higher dimension
   - Adjustment of convergent speed to move faster to the root. Tangent line will be very flat near zero.

   $$x_{n+1} = x_n - \alpha\frac{f(x_n)}{f'(x_n)}$$
6. Newton's method for systems:
   (a) Example: Solve for (x, y) in system

   $$f(x,y) = x^2 + y^2 - 25 = 0, \quad g(x,y) = x^2 - y - 2 = 0$$

   This is the intersection of a circle and parabola.

(b) Formulate system as $\vec{F}(\vec{x}) = \vec{0}$ where $\vec{x} = \langle x, y\rangle$ and $\vec{F} = \langle f, g\rangle$.

(c) Then Newton's method translates to

$$\vec{x}_{n+1} = \vec{x}_n - [\vec{F}'(\vec{x_k})]^{-1}\vec{F}(\vec{x_k})$$

where $J = \vec{F}'$ is the Jacobian matrix defined as

$$J = \vec{F}' = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix}$$

(d) In practice, J^{-1} is not computed. Instead a linear system is solved via GE at each iteration.

$$[\vec{F}'(\vec{x}_k)]H_k = -\vec{F}(\vec{x}_k), \quad \vec{x}_{n+1} = \vec{x}_n + H_k$$

7. Gradient descent method: Instead of solving an equation, we turn to minimizing a function (optimization).

(a) Method:

$$x_{n+1} = x_n - \alpha f'(x_n)$$

(b) Use:

- Finding the global/local minimum of a function (not the zero here). Draw a parabola to give idea.
- Choosing α not too large or small is key. Usually test these by hand.
- Works for high dimensions just the same.
- Optimization problem (least squares, regression, logistic regression, neural network) is the primary application.

(c) Least squares problem and linear regression (Project 2). Motivate with housing example.

- Sample points: (x_i, y_i), find a best fit line $y = h_\theta(x)ax + b$ where h_θ is the hypothesis function and coefficient vector $\theta = \langle a, b\rangle$.
- Minimize

$$J(\vec{\theta}) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(ax_i + b - y_i)^2$$

where $\vec{\theta} = (a, b)$. Will see in project this can be done exactly via projection, but numerically unstable.

- Gradient descent iteration: In general

$$\theta_j^{(n+1)} = \theta_j^{(n)} - \alpha\frac{\partial J}{\partial \theta_j}$$

giving for our 2d case

$$a^{(j+1)} = a^{(j)} - \alpha\sum_{i=1}^{m}(a^{(j)}x_i + b^{(j)} - y_i)x_i$$

$$b^{(j+1)} = b^{(j)} - \alpha\sum_{i=1}^{m}(a^{(j)}x_i + b^{(j)} - y_i)$$

Repeat until convergence. Very simple yet effective enough. Need to monitor the cost function to see that it indeed converges to a minimum. Hard to tell if local / global minimum. Ideal need to show cost function J is convex. Is the case for linear regression.

8. Homework: Exercises 1-4, 6, 9, 17, 23, 30, 32; Computer Ex 1-3, 5, 19, 21, 23, 26; Bonus CEx 27, 28

.3 3.3 Secant method

SKIP SECTION

1. Idea:
   (a) Newton's method
   $$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$
   (b) Replace f' with approximate derivative.
   $$f'(x) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$
   (c) Iteration formula
   $$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$
   Graphical explanation: `https://en.wikipedia.org/wiki/Secant_method`
   (d) Advantage: no longer require f differentiable and need not compute f'.

2. Pseudocode

```
function secant(f, x0, x1, tol)
x = x1;
x1 = x0;
while(|x-x0| > tol)
    x0 = x1;
    x1 = x;
    x = x1 - f(x1) * (x1 - x0) / (f(x1) - f(x0));
end

return x;
```

3. Convergence analysis:

   Theorem .3 (Error estimate of secant method (proof in homework)).
   $$e_{n+1} \approx \frac{1}{2}\frac{f''(r)}{f'(r)}e_n e_{n-1}$$
   Note, not our standard form for convergence rate. Expect slightly worse than quadratic.

4. Comparison with Newton's method
   - Slower but not bad (convergence rate≈ 1.618)
   - Require two starting points (need to be carefully chosen)
   - Don't require knowledge of $f'(x)$
   - Secant method is valid even if $f'(x)$ doesn't exist.

5. Linear interpolation and method of false position
   (a) Idea (homework)
      i. Pick a, b such that $f(a) \cdot f(b) < 0$. Root lies inside $[a, b]$
      ii. Draw the line $l(x)$ connecting $(a, f(a))$ and $(b, f(b))$

iii. Find c such that $l(x) = 0$

iv. Compute $f(c)$

- If $f(c) \cdot f(a) < 0$, pick the subinterval $[a, c]$
- If $f(c) \cdot f(b) < 0$, pick the subinterval $[c, b]$

v. Repeat until convergence

6. Direct iteration (fixed point method)

(a) Solve

$$x - \cos x = 0$$

via

$$x_{n+1} = \cos(x_n)$$

Convergence analysis (homework)

(b) Powerhouse is the following useful theorem.

Theorem .4 (Brouwer's fixed point theorem)**.** *Every continuous function from a closed ball of a Euclidean space into itself has a fixed point.*

(c) Example: if $f(x)$ is a continuous function from $[0, 1]$ to $[0, 1]$, then $f(x)$ has a fixed point in $[0, 1]$, that is

$$x = f(x), \quad \text{for some } x \text{ in } [0, 1].$$

(d) Root finding by Fixed Point theorem

i. Rewrite $f(x) = 0$ to be $g(x) = x$

ii. Prove that $g(x)$ has a fixed point

iii. Define iteration $x_{n+1} = g(x_n)$

iv. Compute til convergence

(e) Nice fact is that I can use fixed point theory to study lots of iterations such as Newton's method.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad \rightarrow \quad x = x - \frac{f(x)}{f'(x)} = g(x)$$

Chapter 4 Interpolation and Numerical Differentiation

Change of topic: Fitting curves to data.

- Interpolation: Given $n + 1$ points, fit a degree n polynomial to those points. Useful for designing / analyzing new methods for calculus problems (num diff, integration)
- Splines: Given a complex-to-evaluate function , replace with a polynomial which approximates on an interval to machine precision.
- Regression (least squares): Given many points, find a best fit polynomial.

.1 4.1 Polynomial Interpolation

1. Interpolation problem: Given $(n + 1)$ points $(x_0, y_0), (x_1, y_1), \ldots (x_n, y_n)$, fit a degree n polynomial to the points such that $p(x_i) = y_i$.
2. Why is polynomial a good choice? Simple, easy to compute and
   - Weierstrauss approximation theorem: Let f be continuous on $[a, b]$. Then for any $\epsilon > 0$, there exists polynomial p such that

$$|f(x) - p(x)| < \epsilon, \quad a \leq x \leq b$$

3. 3+ solutions: All use a different basis to build polynomials.
   - Lagrange form: Write down a formula for p.
   - Newton form: Recursively add points.
   - Linear algebra and the Vandermonde matrix: Solve a linear system.
   - Clever basis: Chebyshev polynomials.
4. Working example: Find a degree two polynomial thru points
$$(0,1), \quad (1,3), \quad (-1,1)$$
5. Lagrange form: Just write down a formula which gives $p(x_i) = y_i$.
   (a) Degree 2 case:
$$p(x) = \sum_{i=0}^{n} \ell_i(x) f(x_i) = \ell_1(x) f(x_1) + \ell_2 f(x_2) + \ell_3 f(x_3)$$
   where $\ell_i(x)$ is the Lagrange basis polynomials (switches) such that
$$\ell_i(x) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$$
   . Try to find formula using working example. Find that
$$\ell_i(x) = \prod \left(\frac{x - x_j}{x_i - x_j} \right).$$
   (b) Visual of Lagrange basis: `https://en.wikipedia.org/wiki/Lagrange_polynomial`
   (c) Drawbacks of Lagrange: Adding a new point hard / need to simplify before evaluation.
6. Newton form: Add one point at a time.
   (a) Example:
      - $p_0(x) = y_0 = 1$
      - $p_1(x) = p_0(x) + c(x - x_0) = -5 + c(x - 0)$. Choose c such that $p_1(x_1) = y_1$.
      - $p_2(x) = p_1(x) + c(x - x_0)(x - x_1)$.
      - Simplify and show same as Lagrange.
      - Add another random term.
      - Note: Newton's form lends itself to Horner nested form for efficient evaluation.
   (b) In general,
$$\begin{aligned} p_n(x) &= \sum_{i=0}^{n} a_i \prod_{j=0}^{i-1} (x - x_j) \\ &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots a_n(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \\ &= p_{n-1}(x) + a_n(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \end{aligned}$$
   (c) Newton divided differences: An efficient way to compute coefficients.
      - Denote the divided difference of order n as $a_n = f[x_0, x_1, \ldots, x_k]$.
      - Write out the polynomial requirements:
        - $f(x_0) = p_n(x_0) = a_0$
        - $f(x_1) = p_n(x_1) = a_0 + a_1(x_1 - x_0)$

- $f(x_2) = p_n(x_2) = a_0 + a_1(x_1 - x_0) + a_2(x_2 - x_0)(x_2 - x_1)$
- $\vdots$

- Algorithm: Solve for each coefficient and rewrite as a divided difference.

$$\begin{aligned}
a_0 &= f[x_0] = f(x_0) \\
a_1 &= f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{f[x_1] - f[x_0]}{x_1 - x_0} \\
a_2 &= f[x_0, x_1, x_2] = \frac{f(x_2) - a_0 - a_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} \\
&= \frac{f[x_2, x_1] - f[x_1, x_0]}{x_2 - x_0} \qquad \cdots \\
a_n &= \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0}
\end{aligned}$$

Note the recursion going on.

- See proof in text. Illustrate formula on previous example. Create a divided difference table.
Theorem .1 (Newton's divided difference formula)**.**

$$f[x_0, x_1, ..., x_k] = \frac{f[x_1, ..., x_k] - f[x_0, ..., x_{k-1}]}{x_k - x_0}$$

  - $f[x_0, x_1, ..., x_n]$: Newton's n^{th} divided difference
  - The order of the sample points doesn't matter
  - Level $n \to n + 1$
  - Connection with slope and linear function.
  - Invariance Theorem: $f[x_0, x_1, \ldots, x_k]$ is invariant under all permutations.

7. Vandermonde Matrix:
   (a) Chose monomials x^n for the basis so that
   $$p_n(x) = c_0 + c_1 x + c_2 x^2 + \ldots c_n x^n$$
   Given $(n + 1)$ points (x_i, y_i) we get a linear system.
   (b) Works well unless n is too large. In which case basis elements x^n become indistinguishable and roundoff error prevails. In short, the Vandermonde matrix becomes ill conditioned for large n.

8. Overview of main approaches:
   - Lagrange: Simple and intuitive. Inefficient computationally. Hard to add / remove points.
   - Newton: Slightly more complex. Computationally efficient. Easy to add points.
   - Vandermonde: Simple. Needs GE to solve. Ill conditioned.

9. Choosing a better polynomial basis: Chebyshev polnomials.
   (a) Polynomial basis on $[-1, 1]$ is derived recursively as
   $$T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x), \quad T_0(x) = 1, \quad T_1(x) = x$$
   (b) Chebyshev polynomials have the advantage of orthogonality (each gives unique information) and range is on $[-1, 1]$. Extrema are equal in magnitude and alternate in sign resulting in a uniform distribution of error in approximation. Chapter 9 discuses further.
   (c) LINK: `https://en.wikipedia.org/wiki/Chebyshev_polynomials`

10. Homework: Exercises 1-5, 7, 8, 12, 42; Computer exercises 1-4, 6,7

.2 4.2 Errors in Polynomial Interpolation

1. Runge's phenomenon:
   (a) Using equally spaced interpolation nodes on Runge's function
   $$f(x) = \frac{1}{1+x^2}, \quad [-5,5]$$
   gives large oscillation for large n.
   (b) Python coding example.
   (c) `https://en.wikipedia.org/wiki/Runge%27s_phenomenon`
2. Remedies for high oscillation:
   (a) Chebyshev nodes:
   - Better choice of node spacing by projecting equadistant points on the upper half of the unit circle onto the x axis.
   - Formula on $[-1,1]$ for $n+1$ nodes:
   $$x_i = \cos\left(\frac{2i+1}{2n+2}\pi\right), \quad 0 \le i \le n$$
   - Formula on $[a,b]$ for $n+1$ nodes:
   $$x_i = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\cos\left(\frac{2i+1}{2n+2}\pi\right), \quad 0 \le i \le n$$
   - Coding example.

   (b) Piece-wise polynomials and splines
3. Theorem: Polynomial interpolation error
   (a) For p_n the degree n polynomial which interpolates f at $(n+1)$ distinct nodes $x_0, x_1, \ldots, x_n$ on interval $[a,b]$, then for each x in $[a,b]$, there exists a ξ in (a,b) such that
   $$f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^{n} (x - x_i)$$
   (b) Proof:
   For $x = x_i$, this works. Assume $x \neq x_i$ and denote
   $$w(x) = (x-x_0)(x-x_1)\cdots(x-x_n)$$
   Consider x as fixed and define
   $$c = \frac{f(x) - p(x)}{w(x)}.$$
   Then, define $g(t)$ in variable t as
   $$g(t) = f(t) - p(t) - cw(t).$$
   Then, g is $(n+1)$ times differentiable since f is and p, w are polynomials. Also,
   $$g(x_i) = 0, \quad g(x) = 0$$
   So g has $(n+2)$ zeros on $[a,b]$. Repeating Rolle's theorem,
   $$g' \quad \text{has } (n+1) \text{ zeros}$$

$$g'' \quad \text{has } n \text{ zeros}$$

$$\vdots$$

$$g^{n+1} \quad \text{has 1 zero on } [a, b]$$

Then,

$$g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - p^{(n+1)}(\xi) - cw^{(n+1)}(\xi) = f^{(n+1)}(\xi) - 0 - c(n+1)! = 0$$

$$\Rightarrow \frac{f(x) - p(x)}{w(x)} = \frac{f^{(n+1)}(\xi)}{(n+1)!}$$

Rearranging, we are done.

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

(c)

Theorem .2 (Rolle's theorem)**.** *If $f(x)$ is continuously differentiable on (a, b) and $f(a) = f(b) = 0$, then there's a ξ in (a, b) such that*

$$f'(\xi) = 0$$

.3 4.3 Estimating Derivatives and Richardson Extrapolation

Chapter 5 Numerical Integration

.1 5.1 Trapezoidal Method

.2 5.2 Romberg Algoritm

.3 5.3 Simpson's Rule and Newton-Cotes Rules

.4 5.4 Gaussian Quadrature Rules

Chapter 6 Spline Functions

.1 6.1 First-Degree and Second-Degree Splines

.2 6.2 Natural Cubic Splines

.3 6.3 B Splines : Interpolation and Approximation

Chapter 7 Initial Value Problems

.1 7.1 Taylor Series Methods

.2 7.2 Runge-Kutta Methods

.3 7.3 Adaptive Runge-Kutta and Multistep Methods

.4 7.4 Methods for First and High-Order Systems

.5 7.5 Adams-Bashforth-Moulton Methods

Chapter 8 More on Linear Systems

.1 8.1

Chapter 9 Least Squares Methods and Fourier Series

.1 9.1

Chapter 6: Solving linear systems

Why do we care? Because of the discreteness of the problems that we see in numerical methods, many need be reframed in the way of linear algebra to solve. This is one

.1 6.1/2 Systems of linear equations

1. Linear system of equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_n \end{aligned}$$

   Matrix equation

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

   where $A_{m\times n} = [a_{ij}]$ is the coefficient matrix, $\vec{x} = [x_i]$ is the vector of unknowns, and $\vec{b} = [b_i]$ is the RHS.

2. Example: Solving a linear system

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 10\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} = \begin{bmatrix}6\\ 15\\ 25\end{bmatrix}$$

3. Review of linear algebra (illustrate with example)

   (a) Augmented matrix

   (b) Elementary row operations (performing these operations to an augmented matrix does not change the solution space)

      i. Add a row to another to replace that row

      ii. Multiply a row by a constant

      iii. Swap two rows

      iv. Row equivalent: Matrices A and B are called row equivalent to each other if A can be transformed into be via only elementary row operations

   (c) Pivot entry (when do we NEED to swap rows?)

   (d) Echelon form

   Theorem .1. *If the augment matrices of two linear systems are row equivalent, then the two linear systems have the same solution set.*

$$\text{System } A \rightarrow \text{elementary row operation} \rightarrow \text{System } B$$

$$\text{Solve System } A \leftrightarrow \text{Solve System } B$$

   - B must be easy to solve.
   - Echelon form makes it easy via backward substitution.

   (e) Solution set (delete a column or row of augmented matrix and discuss possibilities)

   (f) Existence and uniqueness theorem.

   Theorem .2 (Existence of a solution)**.** *The linear system is inconsistent if there is no row of the form*

$$\begin{bmatrix}0 & 0 & \dots & 0 & | & b\end{bmatrix}, \quad b \neq 0$$

   in the echelon form of the augmented matrix.

   Theorem .3 (Uniqueness of a solution)**.** *The solution of a linear system is unique if the coefficient matrix has n pivots.*

   Pivot count is easy to check. Many other conditions for uniqueness of $(n \times n)$ systems:

   - $\det(A) \neq 0$ (show calculation for example here)
   - Column linearly independent
   - Rows linearly independent
   - Range of $T(\vec{x}) = A\vec{x}$ is $\mathbb{R}^n$
   - $A\vec{x} = \vec{0}$ has only the trivial solution.
   - RREF of A is I.
   - A can be written as a product of elementary matrices.
   - More...

4. Numerical method: Gaussian elimination and backwards substitution.

   (a) Gaussian elimination pseudocode: Finding the echelon form of the augmented matrix A

```
for i = 1 : m
    if (all the row below are zero)
        break;
    else
        %eliminate the pivots of row i+1 : m;
        for j = i+1 : m
            row j = row j - row i * A(j,i) / A(i,i);
        end
    end
end
```

(b) Backwards substitution pseudocode: Find the solution of $A\vec{x} = \vec{b}$ where $A_{n\times(n+1)}$ is already an echelon form

```
for i = n : -1 : 1
    x(i) = A(i,n+1)/A(i,i);
    for j = i -1: -1 : 1
        A(j,n+1) = A(j,n+1) - A(j,i) * x(i);
    end
end
```

5. Numerical algorithm summary: Gaussian Elimination + backward substitution
   - It directly mimics hand computation
   - Efficiency: very slow, many operations needed
   - Accuracy: Can be major problems
     - Dividing by zero (easy to avoid)
     - Round off error accumulates

6. Accuracy Issues: Where does the round-off error come from? Two places.

   (a) The system itself is can be ill-conditioned. Famous example is.

$$\text{Hilbert matrix:} \quad H_{ij} = \frac{1}{i+j}$$

   - The condition number of H is huge. Idea of condition number: Given A, let

$$A\vec{x} = \vec{b} \quad \text{and} \quad A\vec{x_*} = \vec{b_*}$$

for $\|\vec{b} - \vec{b}_*\|$ (residual) small. For well-posedness, we hope the inputs $\vec{x}$ and $\vec{x}_*$ are close. Check.

$$A(\vec{x}-\vec{x}_*) = \vec{b}-\vec{b}_* \quad \Rightarrow \quad \|\vec{x}-\vec{x}_*\| \le \|A^{-1}\|\cdot\|\vec{b}-\vec{b}_*\| \quad \Rightarrow \quad \frac{\|\vec{x}-\vec{x}_*\|}{\|\vec{x}\|} \le \|A^{-1}\|\cdot\frac{\|\vec{b}-\vec{b}_*\|}{\|\vec{b}\|}\cdot\frac{\|\vec{b}\|}{\|\vec{x}\|}$$

Note,

$$\frac{\|\vec{b}\|}{\|\vec{x}\|} = \frac{\|A\vec{x}\|}{\|\vec{x}\|} \le \|A\|\frac{\|\vec{x}\|}{\|\vec{x}\|} = \|A\|$$

Finally,

$$\frac{\|\vec{x}-\vec{x}_*\|}{\|\vec{x}\|} \le \|A\|\|A^{-1}\| \cdot \frac{\|\vec{b}-\vec{b}_*\|}{\|\vec{b}\|}$$

and $\text{cond}(A) = \|A\|\|A^{-1}\|$. Note we are looking at relative error here. Also, $\|A\| = \max\|A\vec{x}\|$ for unit vector $\vec{x}$.

   - In short, $\det(A)$ is close to 0

(b) Small pivot combined with larger values. Consider two equivalent linear system

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

i. $\varepsilon = 0$
ii. $\varepsilon =$ 1e-19
iii. Format 2 is better than format 1! Why?
iv. Format 1:

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Echelon form

$$\begin{bmatrix} \varepsilon & 1 \\ 0 & 1 - \frac{1}{\varepsilon} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 - \frac{1}{\varepsilon} \end{bmatrix} \quad \rightarrow \quad \begin{cases} x_2 = \dfrac{2 - 1/\varepsilon}{1 - 1/\varepsilon} \\ x_1 = \dfrac{1 - x_2}{\varepsilon} \end{cases}$$

Consider

$$\frac{1}{\varepsilon} = 10^{19} = 0.1 \cdot 10^{19}, \quad 2 = 0.000\ldots02 * 10^{19}$$

Then

$$2 - 1/\varepsilon \approx -0.1 \cdot 10^{19} = -1/\varepsilon$$

Therefore

$$x_2 = \frac{2 - 1/\varepsilon}{1 - 1/\varepsilon} = \frac{-1/\varepsilon}{-1/\varepsilon} = 1, \quad x_1 = \frac{1 - x_2}{\varepsilon} = 0$$

Number 2 disappeared as a round off error!

(c) Summary

- Gaussian elimination will divide the pivot ε
- If the $\varepsilon = 0$, the process fails
- If the pivot is too small
  - $1/\varepsilon$ is huge
  - other number may disappear as round off errors

Solution: pivoting when small numbers are seen

(d) Partial pivoting

$$\begin{bmatrix} 3 & 1 & -1 & 2 \\ 0 & 0.0001 & 70 & 20 \\ 0 & 10 & 1 & 2 \\ 0 & 30 & 16 & 14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

- Find the largest pivot in the column
- Swap rows 2 & 4
- Swap b values 2 & 4
- Variable order remains

(e) Full pivoting

$$\begin{bmatrix} 3 & 1 & -1 & 2 \\ 0 & 0.0001 & 70 & 20 \\ 0 & 10 & 1 & 2 \\ 0 & 30 & 16 & 14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

- Find the largest pivot in the column & row

- Swap columns 2 & 3
- b values remains
- Swap the variable x_2 and x_3

(f) Gaussian elimination with pivoting summary

- Pivoting solve the roundoff error issue
- Pivoting makes Gaussian elimination even slower
- Pivoting for every step is too much
- Pivoting only when $A(i,i)$ is sufficiently small (or zero)

.2 6.4 Matrix factorization

1. Efficieny drawbacks of Gaussian elimination: Why is it expensive?

   - Backward substitution is cheap
   - Gaussian elimination is expensive

   Improving the efficiency of solving linear system.

   - LU factorization (Package the work for reuse)
   - QR factorization (Make inverting easy)
   - Sparse system (Clever tricks)

2. LU factorization

   (a)

   Theorem .4 (LU factorization)**.** *An LU factorization of A is to rewrite*

   $$A = LU$$

   where

   - *L is lower triangular with diagonal entries all equal to one*
   - *U is upper triangular*

   Solving $A\vec{x} = \vec{b}$

   - Rewrite $A = LU$ (as expensive as Gaussian elimination). Can reuse for many systems.
   - Solve $L\vec{y} = \vec{b}$ (foward substitution), then $U\vec{x} = y$ (backwards substitution) (very cheap)

   (b) Example: Convert to REF.

   $$\left[\begin{array}{ccc|c} 1 & 2 & 3 & 1 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 0 & 2 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 3 & 1 \\ 0 & -3 & -6 & -4 \\ 0 & -6 & -21 & -5 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 3 & 1 \\ 0 & -3 & -6 & -4 \\ 0 & 0 & -9 & 3 \end{array}\right]$$

   (c) Each operation can be caputured as a multiplication by an elementary matrix. Review multiplication, column picture, row picture.

   $$E_1A = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 7 & 8 & 0 \end{bmatrix}$$

   Think row picture here. Combining,

   $$E_3E_2E_1A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -7 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & 0 & -9 \end{bmatrix} = U$$

Inverting elementary matrices is easy. Fill in the blank:

$$E_1^{-1}E_1 = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I$$

Whence,

$$A = (E_3E_2E_1)^{-1}U = E_1^{-1}E_2^{-1}E_3^{-1}U = LU$$

(d) Algorithm

$$L = (E_kE_{k-1}\dots E_2E_1)^{-1}$$

- Make sure L is lower triangular (no row swap)
- The diagonals are 1 (do elimination in a certain order)
- Storage efficiency, place entries of L in the augmented matrix as operations proceed.

$$\left[\begin{array}{ccc|c} 1 & 2 & 3 & 1 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 0 & 2 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 3 & 1 \\ 4 & -3 & -6 & -4 \\ 7 & 2 & -9 & -3 \end{array}\right]$$

(e) Properties of LU factorization

- LU factorization requires pivoting in practice. This amounts to the inclusion of a permutation matrix in the process. $A = PLU$
- L is lower triangular subject to a row swap
- Finding L, U is as expensive as Gaussian elimination
- In practice, it's very common to solve
$$A\vec{x} = \vec{b}$$
with a fixed A and multiple $\vec{b}$

3. QR decomposition

(a) Orthogonal matrix

i. Vectors $\vec{u}$, $\vec{v}$ are orthogonal if $\vec{u}^T\vec{v} = 0$. Think of these two vectors as offering disjoint information.

ii. Orthogonal matrix U

- The columns of U are orthogonal to each other
- The columns of U are unit vectors and so $\|\vec{x}\|^2 = \vec{x}^T\vec{x} = 1$
- If U is an orthogonal matrix, then $U^TU = I$. That is $U^{-1} = U^T$. This is where the advantage lies.

(b) QR decomposition

- Orthogonal matrix Q is constructed via Gram-Schmidt process from the column vectors of A.
- As a result, $A = QR$ is much cheaper than Gaussian elimination, though not as good as LU for solving systems.
- R is an upper triangular matrix.
- Solving
$$A\vec{x} = \vec{b} \quad \Leftrightarrow \quad R\vec{x} = Q^T\vec{b}$$
can be done via backwards substitution.

(c) Idea of QR

i. Refresh basis. Why is orthonormal basis good? Can explicitly rewrite any vector in terms of the basis items. That is, for any $\vec{v} \in \mathbb{V}$ with orthonormal basis $\{\vec{b}_1, \dots\}$

$$\vec{v} = c_1\vec{b}_1 + \dots = \left(\vec{v} \cdot \vec{b}_1\right)\vec{b}_1 + \dots$$

Can show this easily via the dot product and orthogonality and unit basis.

ii. Given matrix A, find an orthonormal basis $\{\vec{b}_1, \vec{b}_2, ..., \vec{b}_n\}$ of Col A

iii. Orthonormal basis of Col A
- Basis $\{\vec{b}_1, \vec{b}_2, ..., \vec{b}_n\}$ spans Col A
- Orthonormal basis
  - $\vec{b}_i^T \vec{b}_j = 0$ for $i \neq j$
  - $\|\vec{b}_i\| = 1$
- Orthonormal matrix
$$Q = [\vec{b}_1 \ \vec{b}_2 \ \vec{b}_3 \ ... \ \vec{b}_n]$$
  Check that $Q^{-1} = Q^T$.

(d) Gram Schmidt process: How to get an orthonormal basis for Col(A)?

i. If the columns of A are linear independent, then the columns $\vec{a}_1$, $\vec{a}_2$, ..., $\vec{a}_n$ forms a basis of Col A.
- Given a basis $\{\vec{a}_1, \vec{a}_2, ..., \vec{a}_n\}$
- Find an orthonormal basis $\{\vec{b}_1, \vec{b}_2, ..., \vec{b}_n\}$

ii. Projection
$$\text{Proj}_{\vec{a}}\vec{b} = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|^2}\vec{b}$$
Draw picture to illustrate. Two vectors not orthogonal. Use the difference to get another which is orthogonal to first and the two span the same plane. Same idea extends to high dimension.

iii. Now, what does our decomposition look like? For each column in A,
$$\begin{aligned}\vec{a}_1 &= \left(\vec{a}_1 \cdot \vec{b}_1\right) b_1 \\ \vec{a}_2 &= \left(\vec{a}_2 \cdot \vec{b}_1\right) b_1 + \left(\vec{a}_2 \cdot \vec{b}_2\right) b_2 \\ &\vdots\end{aligned}$$
Thus $A = QR$.

(e) Gram Schmidt algorithm:
$$\begin{aligned}\vec{b}_1 &= \vec{a}_1 / \|\vec{a}_1\| \\ \vec{x}_{i+1} &= \vec{a}_{i+1} - \sum_{k=1}^{i} (\vec{b}_k^T \vec{a}_{i+1})\vec{b}_k \\ \vec{b}_{i+1} &= \vec{x}_{i+1} / \|\vec{x}_{i+1}\|\end{aligned}$$

(f) Observations:
- Why is R upper triangular? New vectors orthogonal to those prior by construction.
- Twice as expensive as LU factorization
- More stable than LU factorization with pivoting. Major advantage.

(g) More about QR decomposition QR is very handy in
- finding orthonormal basis
- finding the inverse matrix
- eigenvalue theories
- operations in statistics (PCA)
- and many many more...

QR is rated one of the best 10 algorithms in the 20_{th} century

John Francis

4. Sparse matrices
   (a) In practice
      i. Small size system (1000 by 1000)
         - Both QR and Gaussian Elimination work
      ii. Big size system: not doable
      iii. Sparse matrix: A is highly sparse with only a few nonzero entries
         - Visualization
         - Banded system
         - Reordering
         - Gaussian elimination (customized)
         - Block matrices
   (b) Tri-diagonal system example

$$\begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix} \vec{x} = \vec{b}$$

Will see this again in numerical differentiation / integration and differential equations.

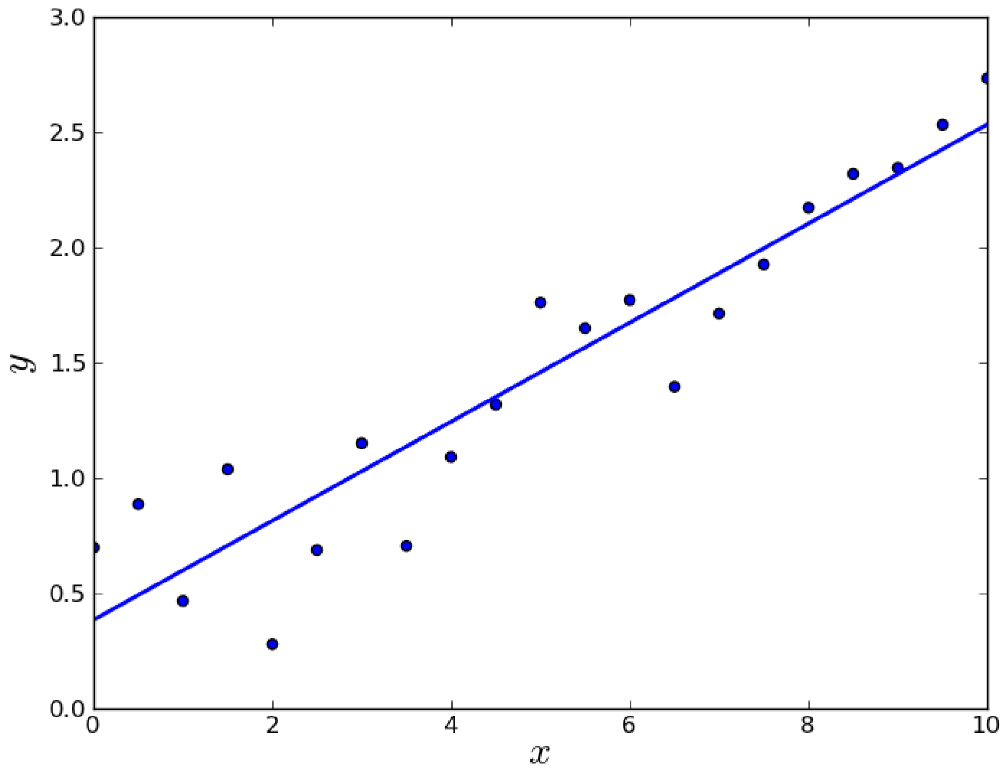
.3 6.* Least Squares

1. Inconsistent system
   (a) Is the following linear system consistent (uniquely solvable)?

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \vec{x} = \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix}$$

Gaussian elimination says no. If we still want a solution, we are going to have to settle for the best approximation.

   (b) A linear system with $m > n$ is in general inconsistent
      - More equation than variables
      - $\vec{b}$ does not belong to Col A
         – $\vec{b}$ is not a linear combination of the columns of A
         – $\vec{b}$ is not in the range of linear transformation $T(\vec{x}) = A\vec{x}$
   (c) Why do we care? Linear regression!



i. Fit a line to data points

$$(x_1, y_1), \quad (x_2, y_2), \quad \ldots, \quad (x_m, y_m)$$

ii. Find $y = ax + b$ such that

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Overdetermined system!

iii. How to solve this overdetermined system? Pick a, b to minimize error.

- Statistics: minimize $\frac{1}{m}\sum_{i=1}^{m}(ax_i + b - y_i)^2$
- Linear algebra: minimize $\|A\vec{x} - \vec{b}\|^2$
- Minimize the difference between prediction and target.

2. Least square approximation process (draw picture to illustrate, summarize process)

$$\text{Find } \vec{x} \text{ such that } A\vec{x} = \vec{b} \text{ where } m \gg n$$

$$\downarrow$$

$$\text{Not solvable}$$

$$\downarrow$$

$$\text{Find } \vec{x} \text{ to minimize } \|A\vec{x} - \vec{b}\|^2$$

$$\downarrow$$

$$\text{Solve } A\vec{x} = \text{Proj}_{\text{Col } A}\vec{b}$$

$$\downarrow$$

$$\text{Find } \vec{x} \text{ such that } A\vec{x} - \vec{b} \text{ is orthogonal to Col } A$$

$$\downarrow$$

$$\text{Find } \vec{x} \text{ such that } A\vec{x} - \vec{b} \text{ is in } (\text{Col } A)^{\perp}$$

3. Column space and null space Let W be a subspace of vector space V
   - $W^{\perp}$: the orthogonal compliment of subspace W.
     - $W^{\perp}$: all vectors in V that are orthogonal to W
     - $W^{\perp}$: all vectors $\vec{x}$ in V satisfying $\vec{x} \cdot \vec{w} = 0$ for any $\vec{w}$ in W.
     - $W^{\perp}$: all vectors $\vec{x}$ in V satisfying $\vec{x} \cdot \vec{w} = 0$ for any $\vec{w}$ in a spanning set of W.
   - (Col A)$^{\perp}$: all vectors $\vec{x}$ in R^n satisfying $\vec{x} \cdot \vec{a}_j = 0$, $1 \leq j \leq n$
   - Null A: the null space of A (the solution space of $A\vec{x} = 0$)
4. A key result on two of the fundamental subspaces of linear algebra.

   Theorem .5. *(Col A)$^{\perp}$ = Null (A^T)*

   Why is this true? Let $\vec{x} \in \text{NULL}(A^T)$. Then $\vec{x} \cdot \vec{a_i} = 0$ for all columns of A.
5. How does this translate to least squares? $A\vec{x} = \vec{b}$ is not solveable. Then, $\vec{b}$ is not in Col(A), but $A\vec{x}$ is. Draw a picture of the column space with $\vec{b}$. So our problems translates.

$$\text{Find } \vec{x}\ A\vec{x} \text{ is closest to } \vec{b}.$$

$$\downarrow$$

$$\text{Find } \vec{x} \text{ such that } A\vec{x} - \vec{b} \in (\text{Col } A)^{\perp}$$
(since the projection is the best approximation)

$$\downarrow$$

$$\text{Find } \vec{x} \text{ such that } A\vec{x} - \vec{b} \in \text{Null } (A^T)$$

$$\downarrow$$

$$\text{Find } \vec{x} \text{ such that } A^T(A\vec{x} - \vec{b}) = 0$$

$$\downarrow$$

$$\text{Find } \vec{x} \text{ which solves the } \textit{normal equations } A^TA\vec{x} = A^T\vec{b} \quad \Rightarrow \quad \vec{x} = (A^TA)^{-1}A\vec{b}$$

Theorem .6. *The least square approximation of $A\vec{x} = \vec{b}$ is the solution of the* **normal equation**

$$A^TA\vec{x} = A^T\vec{b}$$

Why is it called the normal equations? Now we know. Orthogonality.

6. Calculus approach

$$\text{Find } \vec{x} \text{ to minimize } \|A\vec{x} - \vec{b}\|^2$$

$$\downarrow$$

$$\frac{d\|A\vec{x}-\vec{b}\|^2}{d\vec{x}} = 0$$

$$\downarrow$$

$$A^T A\vec{x} = A^T\vec{b}$$

7. QR approach $A = QR$
   - Col A = Col Q
   - $\text{Proj}_{\text{Col }Q}\vec{b} = QQ^T\vec{b}$

$$\text{Solve } A\vec{x} = \text{Proj}_{\text{Col }A}\vec{b}$$

$$\downarrow$$

$$\text{Solve } R\vec{x} = Q^T\vec{b}$$

8. Normal equation $A^T A\vec{x} = A^T\vec{b}$ summary
   - Projection idea
   - Calculus idea
   - n by n linear system
   - Easy to use
   - Not efficient (less than 1000 by 1000)
   - Numerically unstable.
9. General least square problem
   - Multi-variable linear regression
   $$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \vec{x}^T\vec{\theta}$$
   - Polynomial regression
   $$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_n x^n$$
   - $X\vec{\theta} = \vec{y}$
     - Row of X: x values of samples
     - Columns of X: variables
     - $\vec{y}$: y value of samples
     - $\vec{\theta}$: parameter of the model
   - Find θ to minimize
   $$\|X\vec{\theta} - y\|^2, \quad \text{or equivalently} \quad \frac{1}{m}((\vec{x}_i)^T\vec{\theta} - y_i)^2$$
10. Gradient descent method: Define $J(\vec{\theta}) = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}(\theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_n x_n^{(i)} - y_i)^2$

Gradient descent method to minimize $J(\vec{\theta})$

$$\vec{\theta} := \vec{\theta} - \frac{dJ(\vec{\theta})}{d\vec{\theta}}$$

Entrywise (i is variable index, n is number of iteration)

$$\begin{aligned}\theta_j &:= \theta_j - \frac{\partial(J(\vec{\theta}))}{\partial\theta_j} \\ &= \theta_j - (\theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_n x_n^{(i)} - y_i)\cdot x_j^{(i)}\end{aligned}$$

11. Optimization problem: The least ~~square~~ problem

$$\text{Find } \theta \text{ to minimize } J(\theta)$$

- Minimize/Maximize
- Gradient descent
  - Fast
  - Numerical partial derivatives
- Restriction
- Optimizations

12. Optimizations

- $J(\theta)$ = prediction error: regression
- $J(\theta)$ = cost function: machine learning
- $J(\theta)$ = profit: business strategy
- $J(\theta)$ = traveling time: route/schedule making
- $J(\theta)$ = space occupancy: warehouse efficiencey

13. Example:

- Class scheduling
- John Deere intern

Chapter 4: Numerical Interpolation

.1 4.1 Polynomial interpolation

1. Idea:

   (a) Math problem

   $$\text{Given a graph} \rightarrow \text{Produce a formula } f(x)$$

   (b) Numerical problem

   $$\text{Sample points from the real world} \rightarrow \text{Formula } f(x) \text{ (usually a polynomial)}$$

2. Construction: Notation

   - Sample point: (x_i, y_i), $0 \le i \le n$. Total of $(n+1)$ points here.
   - Polynomial: $p(x)$

Theorem .1. *For distinct points x_0, x_1,..., x_n, and y_0, y_1,..., y_n, there's a unique polynomial p of degree $\leq n$ such that*

$$p(x_i) = y_i, \quad 0 \leq i \leq n.$$

3. Finding the interpolating polynomial

   (a) Given distinct points x_0, x_1,..., x_n, and y_0, y_1,..., y_n, find the polynomial p of degree $\leq n$ such that

   $$p(x_i) = y_i, \quad 0 \leq i \leq n.$$

   Let

   $$p(x) = a_0 + a_1 x + a_2 x + \cdots + a_n x^n$$

   Find the coefficients a_i, $0 \leq i \leq n$.

   (b) Why is polynomial a good choice? Simplest idea. Also, it works....

   Theorem .2. *(Weierstrauss Approximation Theorem) Let f be continuous on $[a, b]$. Then for any $\epsilon > 0$, there exists a polynomial $p(x)$ such that*

   $$|f(x) - p(x)| < \epsilon \quad \textit{on [a,b]}$$

   Note, we don't know the degree of said polynomial or how to find it.

   (c) Example: find a degree two polynomial passing through points

   $$(0, 1), \quad (1, 3), \quad (-1, 1)$$

   We will highlight 3 main ways for tackling this problem.

4. **Method 1:** Linear algebra and the Vandermonde matrix.

   (a) Solve $V\vec{x} = \vec{b}$

   $$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & x_1^3 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & \dots & x_n^n \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}}_{\vec{b}}$$

   V: Vandermonde matrix

   (b) Example: find a degree two polynomial passing through points

   $$(0, 1), \quad (1, 3), \quad (-1, 1)$$

   (c) The invertibility of Vandermonde matrix

   i. Is the linear system $V\vec{x} = \vec{b}$ consistent?
      - V: $n + 1 \times n + 1$
      - Is V invertible?

   Theorem .3 (The invertibility of Vandermonde matrix)**.**
   The determinant

   $$\det(V) = \prod_{1 \leq i \neq j \leq n} (x_i - x_j)$$

   - V is invertible if x_0, x_1,. . . , x_n are distinct points.

   (d) Vandemonde matrix is a BAD method
   - We usually take lots of sample points to increase the accuracy.
   - Solving large linear system is expensive.
   - $\det V$ is usually very small (V is ill-conditioned).

- High power result in extremely huge/small matrix entries

5. **Method 2:** Newton interpolating polynomial

   (a) Idea (Newton form): Find a_0, a_1,...,a_n in

   $$\begin{aligned} p(x) =& a_0 + a_1(x-x_0) + a_2[(x-x_0)(x-x_1)] + ... \\ &+ a_n[(x-x_0)(x-x_1)\dots(x-x_{n-1})] \end{aligned}$$

   such that

   $$p(x_i) = y_i, \quad 0 \le i \le n.$$

   Can you see why this works? Why is it constructed this way?

   (b) Example: find a degree two polynomial passing through points

   $$(0,1), \quad (1,3), \quad (-1,1)$$

   Let

   $$p(x) = a_0 + a_1 x + a_2 x(x-1)$$

   Plug in

   $$\begin{aligned} (0,1): &\quad a_0 = 1 \\ (1,3): &\quad a_1 = 2 \\ (-1,1): &\quad a_2 = 1 \end{aligned}$$

   Thus

   $$p(x) = 1 + 2x + x(x-1) = x^2 + x + 1$$

   (c) Algorithm

   $$\begin{aligned} a_0 &= y_0 \\ a_1 &= \frac{y_1 - a_0}{x_1 - x_0} \\ a_2 &= \frac{y_2 - a_0 - a_1(x_2-x_0)}{(x_2-x_0)(x_2-x_1)} \\ a_3 &= \frac{y_3 - a_0 - a_1(x_3-x_0) - a_2(x_3-x_0)(x_3-x_1)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} \\ &\dots \end{aligned}$$

   There appears to be recursion going on. Compactify this schiznit.

   (d) Let $a_n := f[x_0, x_1, x_2, ..., x_n]$, then

   Theorem .4 (Newton's divided difference formula)**.**

   $$f[x_0, x_1, ..., x_k] = \frac{f[x_1, ..., x_k] - f[x_0, ..., x_{k-1}]}{x_k - x_0}$$

   - $f[x_0, x_1, ..., x_n]$: Newton's n^{th} divided difference
   - The order of the sample points doesn't matter
   - Level $n \to n+1$
   - Connection with linear function.

   (e) Divided differences matrix

   $$\begin{bmatrix} D_{11} & 0 & \dots & \dots & 0 \\ D_{21} & D_{22} & \dots & \dots & 0 \\ \vdots & \vdots & & & \vdots \\ D_{n1} & D_{n2} & \dots & \dots & D_{nn} \end{bmatrix}$$

- D is $n+1$ by $n+1$.
- D is lower triangular.
- $a_i = D(i+1, i+1)$, for $0 \le i \le n$.

(f) Algorithm: Construct D (x, y are the columns vectors of the data points)

```
D(:,1) = y;
for j = 2 : n
    for i = j : n
        D(i,j)=(D(i,j-1)-D(i-1,j-1))/(x(i)-x(i-j+1));
    end
end
```

(g) Rewrite above example in this format.

6. **Method 3** Lagrange polynomial

(a) Idea of Lagrange (Cardinal functions): Given x_0, x_1, ..., x_n, the cardinal functions l_0, l_1, ..., l_n are degree n polynomials satisfying

$$l_i(x_j) = \delta_{ij} := \begin{cases} 0, & i \ne j, \\ 1, & i = j. \end{cases}$$

Thus the interpolating polynomial

$$p(x) = \sum_{i=0}^{n} l_i(x) y_i$$

Thus,

$$l_i(x) = \prod_{j=0,\ j\ne i}^{n} \frac{x - x_j}{x_i - x_j}$$

(b) Example: use Lagrange interpolating polynomial to find a degree two polynomial passing through points

$$(0,1), \quad (1,3), \quad (-1,1)$$

Find

$$l_0(x) = \frac{(x-1)(x+1)}{(0-1)(0+1)} = 1 - x^2$$
$$l_1(x) = \frac{x(x+1)}{(1-0)(1+1)} = \frac{x(x+1)}{2}$$
$$l_2(x) = \frac{x(x-1)}{(-1-0)(-1-1)} = \frac{x(x-1)}{2}$$

Then

$$p(x) = \sum_{i=0}^{2} l_i(x) y_i = l_0(x) + 3l_1(x) + l_2(x) = x^2 + x + 1$$

.2 4.2 Numerical analysis of polynomial interpolation

1. Comparison:

(a) Vandermonde matrix method (we should avoid this method)

- Nice structure
- Expensive to compute
- The matrix is usually ill-conditioned

(b) Newton interpolating method (most practical)
   - Easy to code
   - The most efficient

(c) Lagrange interpolating method
   - Easy to write down (theoretical uses)
   - Not so efficient ($\mathcal{O}(n^2)$)

2. Error estimate

   (a)

   Theorem .5 (Error estimate for polynomial interpolation). *Let p be the interpolating polynomial of $f(x)$ at $x_0, \ldots, x_n$, which belongs to an interval $[a,b]$. If $f^{(n+1)}(x)$ is continuous, then there is a ξ in (a,b) for which*

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^{n} (x - x_i), \quad a \le x \le b.$$

   (b) Proof:
   For $x = x_i$, this works. Assume $x \neq x_i$ and denote

$$w(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$$

   Consider x as fixed and define

$$c = \frac{f(x) - p(x)}{w(x)}.$$

   Then, define $g(t)$ in variable t as

$$g(t) = f(t) - p(t) - cw(t).$$

   Then, g is $(n+1)$ times differentiable since f is and p, w are polynomials. Also,

$$g(x_i) = 0, \quad g(x) = 0$$

   So g has $(n+2)$ zeros on $[a,b]$. Repeating Rolle's theorem,

$$g' \quad \text{has } (n+1) \text{ zeros}$$

$$g'' \quad \text{has } n \text{ zeros}$$

$$\vdots$$

$$g^{n+1} \quad \text{has 1 zero on } [a,b]$$

   Then,

$$g^{(n+1)}(\xi) = f^{(n+1)}(\xi) - p^{(n+1)}(\xi) - cw^{(n+1)}(\xi) = f^{(n+1)}(\xi) - 0 - c(n+1)! = 0$$

$$\Rightarrow \frac{f(x) - p(x)}{w(x)} = \frac{f^{(n+1)}(\xi)}{(n+1)!}$$

   Rearranging, we are done.

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

(c)

Theorem .6 (Rolle's theorem)**.** *If $f(x)$ is continuously differentiable on (a,b) and $f(a) = f(b) = 0$, then there's a ξ in (a,b) such that*

$$f'(\xi) = 0$$

3. The convergence analysis
   - Require $f^{(i)}(x)$ is continuous for $0 \le i \le n+1$
   - More sample points $\rightarrow$ better accuracy?

   $$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x - x_i)$$

   - In general, higher order (more sample points) does not guaranteed better accuracy!!

   $$f^{(n+1)}(\xi) \quad \text{can grow fast as } n \text{ increases}$$

4. Runge function:

   $$f(x) = \frac{1}{1 + 25x^2}$$

   - $f(x)$ is infinitely many differentiable on $(-1,1)$
   - Find the interpolating polynomial on $[-1,1]$
   - Use sample points with equal distance

   The error blow up because

   $$f^{(n+1)}(\xi) \quad \text{grow very fast as } n \text{ increases}$$

5. Cure for the high error:
   - Choose sample points carefully: Chebyshev nodes
   - Remove the requirement

     $$p(x_i) = y_i$$

     Regression (least square)
   - Use many low degree interpolation instead of one high degree.
6. Comparison with interpolation and regression
   - High order regression
   - Interpolation is an "overfitting" regression
   - Train error vs prediction error
     - Training: sample points
     - Prediction: evaluation points
   - Memorizing too much won't let you learn.
7. Evaluating polynomial interpolation
   - Test function $f(x)$
     - Standard ones
     - Good ones
     - Bad ones
   - Evaluation points $(x_i, f(x_i))$: different from the sample points

- Homework

8. Idea behind the interpolation polynomial
   Vector space $V :=$ all infinitely continuous differentiable functions

   - dim $V = \infty$
   - A basis of V:
   $$1, \quad x, \quad x^2, \quad x^3, \quad ... \quad x^n \quad ...$$
   - Then any $f(x)$ in V can be written as a linear combination
   $$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$
   - Not doable.

9. Idea behind interpolation polynomial

   - Subspace $H =$ span $\{1, x, x^2, ..., x_n\}$
   - Projection of f
   $$\bar{f} = Proj_H f$$
   - Find the linear combination
   $$\bar{f} = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$
   - Error $f - \bar{f}$ is determined by the subspace H.

10. Wrap up
    Infinitely dimensional V – Basis B: infinitely many vectors

    Find the linear combination of f in V using the basis

    Subspace H – Basis: a subset of B

    Find the linear combination of $\bar{f}$ in H using the basis

    where $\bar{f}$ is the projection of f on H.

11. Fourier transform:
    Signal space V: all period functions (formally)

    - Basis
    $$1, \quad \sin(x), \quad \cos(x), \quad \sin(2x), \quad \cos(2x), \quad \ldots$$
    - Find the linear combination of f in V such that
    $$f = C + a_1 \sin(x) + b_1 \cos(x) + a_2 \sin(2x) + b_2 \cos(2x) + \cdots$$

    The Fourier Series `https://en.wikipedia.org/wiki/Doppler_spectroscopy`
    `https://www.kaggle.com/mrisdal/open-exoplanet-catalogue`

.3 4.3 Splines

1. Goal: Reduce the interpolating error. How?
   (a) ~~More sample points (higher order)~~ (issue is with equally spaced nodes)
   (b) Choose sample points carefully (Chebyshev)
   (c) The spline function: piecewise low degree interpolation

   Matlab example to illustrate these three.

2. Linear spline function: The linear spline function of $f(x)$ on $[a, b]$
   (a) Draw picture to illustrate.
   (b) Knots: sample points on interval $[a, b]$

   $$x_0, x_1, \ldots, x_n, \quad \text{where} \quad x_0 = a, \quad x_n = b$$

   (c) Partition of [a,b]: $[x_{i-1}, x_i], \quad 1 \le i \le n$
   (d) In each interval, define a linear function $p_i(x)$ such that

   $$p_i(x_{i-1}) = f(x_{i-1}), \quad p_i(x_i) = f(x_i), \quad 1 \le i \le n$$

   (e) Linear piece: for $1 \le i \le n$

   $$p_i(x) = f(x_{i-1})\frac{x - x_i}{x_{i-1} - x_i} + f(x_i)\frac{x - x_{i-1}}{x_i - x_{i-1}}$$

   (f) Linear spline function

   $$p(x) = p_i(x), \quad \text{for} \quad x \in [x_{i-1}, x_i), \quad 1 \le i \le n$$

   (g) If we have a *uniform mesh*, rewrite with $h = \Delta x = x_i - x_{i=1}$.

3. Advantages/disadvantages of Linear spline function

   $$\text{One high order } p(x) \quad \textbf{VS} \quad \text{multiple lower order } p_i(x)$$

   - Adv: More sample points $\rightarrow$ better accuracy
   - Adv: Efficient computation
   - Adv: Works for weird shape (non-functionss)
   - Dis: Sacrifice smoothness
     - Ultimate solution: Higher order spline functions (cubic is king)

4. Spline function definition: A function $S(x)$ is called a spline of degree k on $[a, b]$ if
   - Domain of S: $[a, b]$
   - Knots: $a = x_0 < x_1, \cdots < x_n = b$
   - S is a polynomial of degree less or equal to k on each subinterval $[x_i, x_{i-1}]$.
   - S', S'',. . . , $S^{(k-1)}$ is continuous

   Linear spline function is the degree 1 spline function.

5. Degree 2 spline function
   (a) Continuity

   $$S_i(x_{i-1}) = f(x_{i-1}), \quad S_i(x_i) = f(x_i), \quad 1 \le i \le n$$

(b) Continuity of $S'(x)$ on interior nodes only.

$$S'_i(x_i) = S'_{i+1}(x_i), \quad 1 \le i \le n-1$$

- No. of variables: $3n$
- No. of equations: $n + n + n - 1 = 3n - 1$
- Manually add: $S'(x_0) = f'(x_0)$

6. Cubic spline function: Degree 3 spline function satisfies

$$S_i(x_{i-1}) = f(x_{i-1}), \quad S_i(x_i) = f(x_i), \quad 1 \le i \le n$$
$$S'_i(x_i) = S'_{i+1}(x_i), \quad S''_i(x_i) = S''_{i+1}(x_i), \quad 1 \le i \le n-1$$

- No. of variables: $4n$
- No. of equations: $4n - 2$ (Nice added flexibility here.)
- Manually add 2 equations: Many choices here. What is the interpretation of these conditions?
  - Natural cubic spline:
  $$S''_1(x_0) = S''_n(x_n) = 0$$
  - Complete cubic spline:
  $$S'_1(x_0) = f'(x_0), \quad S'_n(x_n) = f'(x_n)$$
- First derivative: increasing/decreasing
- Second derivative: concavity/curvature

7. Example: Find the natural cubic spline function through knots

$$(-1, 1), \quad (0, 2), \quad (1, -1)$$

(a) Define

$$S_1(x) = a_1x^3 + b_1x^2 + c_1x + d_1$$
$$S_2(x) = a_2x^3 + b_2x^2 + c_2x + d_2$$

(b) Need 8 equations!

$$\begin{cases} S_1(-1) = 1, & S_1(0) = 2, & S_2(0) = 2, & S_2(1) = -1, \\ S'_1(0) = S'_2(0), & S''_1(0) = S''_2(0), & S''_1(-1) = 0, & S''_2(1) = 0 \end{cases},$$

that is

$$\begin{cases} a_1(-1)^3 + b_1(-1)^2 + c_1(-1) + d_1 = 1 \\ a_1(0)^3 + b_1(0)^2 + c_1(0) + d_1 = 2 \\ a_2(0)^3 + b_2(0)^2 + c_2(0) + d_2 = 2 \\ a_2(1)^3 + b_2(1)^2 + c_2(1) + d_2 = -1 \\ 3a_1(0)^2 + 2b_1(0) + c_1 = 3a_2(0)^2 + 2b_2(0) + c_2 \\ 6a_1(0) + 2b_1 = 6a_2(0) + 2b_2 \\ 6a_1(-1) + 2b_1 = 0 \\ 6a_2(1) + 2b_2 = 0 \end{cases}$$

(c) Solution

$$a_1 = -1, \quad b_1 = -3, \quad c_1 = -1, \quad d_1 = 2$$
$$a_2 = 1, \quad b_2 = -3, \quad c_2 = -1, \quad d_2 = 2$$

thus the natural spline function

$$S(x) = -x^3 - 3x^2 - x + 2, \quad -1 \le x \le 0$$
$$S(x) = x^3 - 3x^2 - x + 2, \quad 0 \le x \le 1$$

We definitely need a better algorithm!

8. Algorithm: For finding any cubic spline.

(a) Goal: For a given function f and $(n+1)$ nodes x_i,

$$S_i(x_{i-1}) = f(x_{i-1}), \quad S_i(x_i) = f(x_i), \quad 1 \le i \le n$$

$$S_i'(x_i) = S_{i+1}'(x_i), \quad S_i''(x_i) = S_{i+1}''(x_i), \quad 1 \le i \le n-1$$

Check off these conditions as we go.

(b) Assign

$$z_i = S''(x_i), \quad 1 \le i \le n-1$$

Since $S(x)$ is cubic, $S''(x)$ is linear. Therefore

$$S_i''(x) = z_{i-1}\frac{x - x_i}{x_{i-1} - x_i} + z_i \frac{x - x_{i-1}}{x_i - x_{i-1}}, \quad x_{i-1} \le x \le x_i$$

Assume uniform partition with step h

- $x_i - x_{i-1} = h$ for $1 \le i \le n$
- $x_i = x_0 + ih$ for $1 \le i \le n$

(c) Then on interval $[x_{i-1}, x_i]$

$$S_i''(x) = z_{i-1}\frac{x_i - x}{h} + z_i \frac{x - x_{i-1}}{h},$$

thus

$$S_i'(x) = -\frac{1}{h} z_{i-1}\frac{(x_i - x)^2}{2} + \frac{1}{h} z_i \frac{(x - x_{i-1})^2}{2} + C_i,$$

Finally for some constant C_i, D_i

$$S_i(x) = \frac{1}{h} z_{i-1}\frac{(x_i - x)^3}{6} + \frac{1}{h} z_i \frac{(x - x_{i-1})^3}{6} + C_i(x - x_{i-1}) + D_i$$

Noticed we borrowed some Newton divided difference cleverness here.

(d) Find z_i's, C_i and D_i in

$$S_i(x) = \frac{1}{h} z_{i-1}\frac{(x_i - x)^3}{6} + \frac{1}{h} z_i \frac{(x - x_{i-1})^3}{6} + C_i(x - x_{i-1}) + D_i$$

- Condition (rewrite $f(x_i)$ as y_i)

$$S_i(x_{i-1}) = f(x_{i-1}) \Rightarrow D_i = y_{i-1} - \frac{h^2}{6} z_{i-1}$$

- Condition

$$S_i(x_i) = f(x_i) \Rightarrow C_i = \frac{1}{h}\Big[y_i - y_{i-1} + \frac{h^2}{6}(z_{i-1} - z_i)\Big)\Big]$$

(e) Plug C_i and D_i in

$$\begin{aligned} S_i(x) =& \frac{1}{h} z_{i-1}\frac{(x_i - x)^3}{6} + \frac{1}{h} z_i \frac{(x - x_{i-1})^3}{6} \\ &+ \frac{1}{h}\Big[y_i - y_{i-1} + \frac{h^2}{6}(z_{i-1} - z_i)\Big)\Big](x - x_{i-1}) \\ &+ y_{i-1} - \frac{h^2}{6} z_{i-1} \end{aligned}$$

(f) It remains to find z_i for $1 \le i \le n-1$. Recall:

$$S_i'(x) = -\frac{1}{h} z_{i-1} \frac{(x_i - x)^2}{2} + \frac{1}{h} z_i \frac{(x - x_{i-1})^2}{2} + C_i,$$

For $0 \le i \le n-1$, condition

$$S_i'(x_i) = S_{i+1}'(x_i) \Rightarrow \frac{h}{2} z_i + C_i = -\frac{h}{2} z_i + C_{i+1}$$

Plug in C_i, C_{i+1}

$$\frac{h}{2} z_i + \frac{1}{h}\Big[y_i - y_{i-1} + \frac{h^2}{6}(z_{i-1} - z_i))\Big]$$
$$= -\frac{h}{2} z_i + \frac{1}{h}\Big[y_{i+1} - y_i + \frac{h^2}{6}(z_i - z_{i+1}))\Big]$$

(g) Finding z_i: Organize the terms

$$\frac{h}{6} z_{i-1} + \frac{2h}{3} z_i + \frac{h}{6} z_{i+1} = \frac{1}{h} y_{i-1} - \frac{2}{h} y_i + \frac{1}{h} y_{i+1}, \quad 1 \le i \le n-1$$

Notice that

$$z_0 = z_n = 0$$

Therefore we can rewrite the equations into a linear system

(h) Solve $[z_1, z_2, \ldots, z_{n-1}]^T$ from

$$\begin{bmatrix} 2h/3 & h/6 & 0 & \ldots & \ldots & 0 \\ h/6 & 2h/3 & h/6 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & 0 & h/6 & 2h/3 & h/6 \\ 0 & \ldots & \ldots & 0 & h/6 & 2h/3 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix}$$

where

$$b_i = \frac{1}{h}\Big(y_{i-1} - 2y_i + y_{i+1}\Big), \quad 1 \le i \le n-1$$

- The coefficient matrix is $n-1$ by $n-1$ and invertible
- The coefficient matrix is a tri-diagonal matrix
- For non-uniform partition, use h_i instead of h

9. General curve fitting
   - Non-function
   - Data fitting
   - 3-D curve fitting
   - 3-D spline plane

10. Spline function: Advantage
    - Low order polynomial
    - More sample points implies better accuracy
    - Non-function
    - Data interpolation
    - Works for 3-D

- Surface fitting

11. Error analysis: Error estimate for polynomial interpolation

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{i=0}^{n} (x - x_i), \quad a \le x \le b.$$

Error analysis for spline function

$$E = \max_{1 \le i \le n} e_i$$

where e_i is the same error form on $[x_{i-1}, x_i]$.

Why is spline function better?

Chapter 5: Numerical Differentiation and Integration

.1 5.1 Numerical Differentiation

1. Idea: Given sample points of $f(x)$, approximate the derivative $f'(x)$. Draw picture to illustrate.
   (a) Interpolation is not a good choice here. Not efficient, hard to measure accuracy.
   (b) Ideas from calculus:
      - Difference quotient (look familiar?)
      $$f'(x_n) \approx \frac{f(x_{n+1}) - f(x_n)}{x_{n+1} - x_n}$$
      - Central difference (which is better?)
      $$f'(x_n) \approx \frac{f(x_{n+1}) - f(x_{n-1})}{x_{n+1} - x_{n-1}}$$
   (c) Consider uniformly distributed sample points ($h = x_i - x_{i-1}$ for all i).
      - Difference quotient
      $$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$
      - Central difference
      $$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$
      - Which is better? Central difference! Key to concrete comparison is Taylor series.
2. First derivative approximation: $f'(x_i) = ?$.
   (a) Formula 1: Taylor series (for some $x < \xi < x + h$)
   $$f(x+h) = f(x) + f'(x)h + \frac{f''(\xi)}{2} h^2$$
   thus
   $$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{1}{2} h f''(\xi) \approx \frac{f(x+h) - f(x)}{h}$$
      - Truncation error (assuming $f''(x)$ is bounded)
   $$E = |-\frac{1}{2} h f''(\xi)| \le \frac{1}{2} h \max_{x < \xi < x+h} |f''(\xi)| \le Ch \sim O(h)$$

- Convergence rate

$$O(h^k)\text{: order } k \text{ convergence}$$

(b) Example: Approximate the derivative of $y = e^x$ at 0

$$f'(0) \approx \frac{e^{0+h} - e^0}{h}$$

h	0.1	0.01	0.001
error	0.05	0.005	5e-04

(c) Formula 2: Taylor series (for some $x < \xi < x + h$ and $x - h < \eta < x$)

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(\xi)$$
$$f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(\eta)$$

Take subtraction we see nice canceling.

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{1}{6}h^3([f'''(\xi) + f'''(\eta)]$$

Then,

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{6}h^2[f'''(\xi) + f'''(\eta)] \approx \frac{f(x+h) - f(x-h)}{2h}$$

Truncation error: (assuming $f'''(x)$ is bounded)

$$E = \left| -\frac{1}{6}h^2[f'''(\xi) + f'''(\eta)] \right| \leq \frac{1}{6}h^2 \cdot 2 \max_{x-h<\xi<x+h} |f^{(3)}(\xi)| \sim O(h^2)$$

(d) Example: Approximate the derivative of $y = e^x$ at 0

$$f'(0) \approx \frac{e^{0+h} - e^{0-h}}{2h}$$

h	0.1	0.01	0.001
error	1.67e-03	1.67e-05	1.67e-07

(e) Super convergence: Approximate the derivative of $y = \sin x$ at 0

$$f'(0) \approx \frac{\sin(0+h) - \sin 0}{h}$$

h	0.1	0.01	0.001
error	1.67e-03	1.67e-05	1.67e-07

Why order 2?

Super convergence!

Why? Think of the Taylor series for sine in above argument. Only odd powers.

(f) High order of convergence: What if I want order 3 convergence?

Undetermined coefficient

$$f'(x) = \frac{[f(x+h) - \frac{1}{3}f(x-h) - \frac{1}{6}f(x+2h) - \frac{1}{2}f(x)]}{h^2} + O(h^3)$$

- High order derivative
- High order of convergence
- Picking the point smartly to get even higher order convergence. Construct above by combining $Af(x+h), Bf(x-h), Cf(x+2h)$ and solving for A, B, C to eliminate low order terms.
- Coding experiment to illustrate above. Do more calculation $k = 10$ to show roundoff error taking over. WTF?

(g) Round-off error concerns.

i. For the above finite differences, the error gets smaller at first, then error grows unexpectedly. This is because of roundoff error. Division by small h contributes here.

ii. Approximation: Goal is

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

Here is what the machine does. $f(x+h)$ is rounded to $f(x+h)(1+\delta_1)$ and $f(x)$ to $f(x)(1+\delta_2)$ where $|\delta_1|, |\delta_2| < \epsilon$ (machine accuracy). Then considering machine and truncation error, note the equality below.

$$\begin{aligned} f'(x) &= \frac{f(x+h)(1+\delta_1) - f(x)(1+\delta_2)}{h} - \frac{h}{2}f''(\xi) \\ &= \frac{f(x+h) - f(x)}{h} + \frac{\delta_1 f(x+h) - \delta_2 f(x)}{h} - \frac{h}{2}f''(\xi) \\ &\leq \frac{f(x+h) - f(x)}{h} + \epsilon\frac{|f(x+h)| + |f(x)|}{h} - \frac{h}{2}f''(\xi) \end{aligned}$$

True error in red. Note, the first increases as h decreases and the second does opposite. We need to balance these two. So minimize $\frac{\epsilon}{h} + h$.

$$\frac{d}{dh}\left(\frac{\epsilon}{h} + h\right) = 0 \Rightarrow h = \sqrt{\epsilon} \approx 10^{-8}$$

So we can only expect 8 digits accuracy.

(h) Summary: Numerical approximation of the derivative

- Taylor expansion
- Truncation error
- Various formulas (many different ways)
- Higher order possible if carefully constructed.
- Take care with roundoff error.

3. Second derivative: $f''(x) = ?$

(a) Taylor series (for some $x < \xi < x+h$ and $x - h < \eta < x$)

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + \frac{1}{24}h^4 f^{(4)}(\xi) \\ f(x-h) &= f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(x) + \frac{1}{24}h^4 f^{(4)}(\eta) \end{aligned}$$

Take the sum

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + O(h^4)$$

then

$$f''(x) = \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} + O(h^2)$$

(b) Example: Approximate the second derivative of $y = e^x$ at 0

$$f'(0) \approx \frac{e^{0+h} + e^{0-h} - 2e^0}{h^2}$$

h	0.1	0.01	0.001
error	8.3e-04	8.3e-06	8.3e-08

4. Richardson's extrapolation

(a) Idea:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f'''(x) - \frac{h^4}{120} f^{(5)}(x) + \cdots$$

Eliminate the red term.

(b) Define:

$$\Phi_0(h) = \frac{f(x+h) - f(x-h)}{2h}$$

Consider

$$f'(x) = \Phi_0(h) - \frac{h^2}{6} f'''(x) + O(h^4) \qquad (1)$$

$$f'(x) = \Phi_0\left(\frac{h}{2}\right) - \frac{h^2}{24} f'''(x) + O(h^4) \qquad (2)$$

Then $4 \cdot (2) - (1)$

$$3f'(x) = 4\Phi_0\left(\frac{h}{2}\right) - \Phi_0(h) + O(h^4)$$

Thus

$$f'(x) = \frac{4}{3}\Phi_0\left(\frac{h}{2}\right) - \frac{1}{3}\Phi_0(h) + O(h^4)$$

Formula

$$f'(x) = \frac{4f(x+\frac{h}{2}) - 4f(x-\frac{h}{2})}{3h} - \frac{f(x+h) - f(x-h)}{6h}$$

with truncation error $O(h^4)$

(c) Even higher order

$$f'(x) = \frac{16}{15}\Phi_1\left(\frac{h}{2}\right) - \frac{1}{15}\Phi_1(h) + O(h^6)$$

where

$$\Phi_1(h) = \frac{4}{3}\Phi_0\left(\frac{h}{2}\right) - \frac{1}{3}\Phi_0(h)$$

Balance between

- High order error estimate
- Computing cost of Φ

(d) Summary: `https://en.wikipedia.org/wiki/Richardson_extrapolation`

i. Easy to use: recursive

ii. Higher order VS smaller h: Let $h = 0.1$ on $[0, 1]$

- Central difference: $O(h^2)$

- 10 sample points
- Errror: 0.01 level

- One step Richardson: $O(h^4)$
  - 20 sample points
  - Error: 0.0001 level

iii. Machine error: high order convergence uses bigger h

.2 5.2 Definite integral and the Trapezoid Rule

1. Champion of the calculus:

   Theorem .1 (Fundamental theorem of calculus)**.** *If $f(x)$ is continuous on $[a,b]$, then the definite integral*

   $$\int_a^b f(x)\,dx = F(b) - F(a) \quad \textit{where} \quad F'(x) = f(x)$$

   - Monumental: connects 2 parts of calculus (differentiation and area under curve)
   - Practical: area under a curve
   - Calculus is not practical: finding $F(x)$ (antiderivative) is hard

   $$\int_0^4 e^{-x^2}\,dx = ?$$

   - Solution: numerical integration (approximation)

2. Numerical integration: Interpolation idea.

   (a) Straightforward (but badish) idea is to replace $f(x)$ with a polynomial interpolant.
   - Define partition nodes
     $$a = x_0 < x_1 < x_2 \cdots < x_n = b$$
     Easiest to choose uniform, $h = \frac{b-a}{n}$.
   - Find the polynomial interpolation $p(x)$
   - Find the definite integral of $p(x)$ instead
   - Disadvantage:
     - Not efficient
     - Unstable (Runge's phenomenon)

   (b) Newton-Cotes rules
   - Define equal space nodes
     $$x_i = a + ih, \quad 0 \le i \le n$$
   - Find the Lagrange Interpolation $p(x)$
     $$p(x) = \sum_{i=0}^{n} f(x_i) \prod_{j=0,\ j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$
   - Find the definite integral of $p(x)$ instead
     $$\int_a^b f(x)\,dx = \sum_{i=0}^{n} f(x_i) \int_a^b \prod_{j=0,\ j\neq i}^{n} \frac{x - x_j}{x_i - x_j}\,dx$$
   - Compute exactly for $n = 1, 2$.

- Get the error estimate from polynomial interpolation formula.

3. Numerical integral: Better idea (as we saw for interpolation) is to do above peicewise.
   (a) Partition the domain into subintervals (uniform size easiest).
   - Partition
   $$a = x_0 < x_1 < \cdots < x_n = b$$
   - Find spline function $S(x)$ on each subinterval.
   - Find the integral of $S(x)$ instead.

   (b) This idea is not new. Riemann sum is the degree 0 case.
   $$\int_a^b f(x)\ dx = \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i^*) \cdot \frac{b-a}{n}$$
   - Partition of $[a, b]$: n uniform intervals
   $$[x_0, x_1], \quad [x_1, x_2], \quad \ldots, \quad [x_{n-1}, x_n]$$
   - x_i^*: any number in interval $[x_{i-1}, x_i]$
   - Left end formula: $x_i^* = x_{i-1}$
   - Right end formula: $x_i^* = x_i$
   - Mid points formula: $x_i^* = (x_{i-1} + x_i)/2$

   (c) Rectanglular rule: Degree 0 polynomial on subinterval.
   - Left point rule with node $x = x_{i-1}$
   $$\int_a^b f(x)\ dx \approx \sum_{i=1}^{n} f(x_{i-1})(x_i - x_{i-1})$$
   - Right point rule with node $x = x_i$
   $$\int_a^b f(x)\ dx \approx \sum_{i=1}^{n} f(x_i)(x_i - x_{i-1})$$
   - Midpoint rule with node $x = (x_{i-1} + x_i)/2$
   $$\int_a^b f(x)\ dx \approx \sum_{i=1}^{n} f\Big(\frac{x_{i-1} + x_i}{2}\Big)(x_i - x_{i-1})$$

   (d) Trapezoid rule: Linear polynomial on subinterval.
   - Nodes: x_{i-1} and x_i both used.
   - Trapezoid rule
   $$\int_a^b f(x)\ dx \approx \sum_{i=1}^{n} \frac{1}{2}\Big(f(x_{i-1}) + f(x_i)\Big)(x_i - x_{i-1})$$

   (e) Example
   - Use 4 different formula to compute
   $$\int_0^2 e^{-x^2}\ dx$$
   - Matlab example
   - Left/right formula: $O(h)$
   - Midpoint/Trapezoid: $O(h^2)$

- Mid point error = 1/2 Trapezoid error
- How to see rate of convergence? Create a log of errors halfing the stepsize at each step.

$$\frac{e_h}{e_{h/2}} = \frac{\mathcal{O}(h^p)}{\mathcal{O}((h/2)^p)} \approx 2^p$$

Then, for h small,

$$p \approx \log_2\left(\frac{e_h}{e_{h/2}}\right)$$

Add this to sample code.

4. Error analysis

   (a)

   Theorem .2 (Fundamental theorem of calculus (second)). *If $f(x)$ is continuous on (a, b) and let*

$$F(x) = \int_a^x f(t)\ dt$$

   Then

$$F'(x) = f(x), \quad a < x < b$$

   (b) Error analysis for left endpoint rule: Battle is we need to connect the integral of f to f. Start with the Taylor series of antiderivative F.

   i. For interval $[x_i, x_{i+1}]$

   - Taylor series of $F(x)$ at x_i

$$\begin{aligned} F(x_i + h) &= F(x_i) + hF'(x_i) + \frac{1}{2}h^2F''(x_i) + \dots \\ &= F(x_i) + hf(x_i) + \frac{1}{2}h^2f'(\xi_i) \end{aligned}$$

   - Therefore for some ξ in $[x_i, x_{i+1}]$

$$\int_{x_i}^{x_{i+1}} f(x)\ dx = F(x_i + h) - F(x_i) = hf(x_i) + \frac{1}{2}h^2f'(\xi_i)$$

   Error for left point formula on the i_{th} interval

$$E_i \sim O(h^2)$$

   ii. Error analysis: Numerical integral

$$\begin{aligned} \int_a^b f(x)\ dx &= \sum_{i=1}^n \int_{x_i}^{x_{i+1}} f(x)\ dx \\ &= \sum_{i=1}^n hf(x_i) + \frac{1}{2}h^2\Big(f'(\xi_1) + f'(\xi_2) + \dots + f'(\xi_n)\Big) \end{aligned}$$

   Total error

$$\begin{aligned} E &= \frac{1}{2}h^2\Big(f'(\xi_1) + f'(\xi_2) + \dots + f'(\xi_n)\Big) \\ &\le \frac{1}{2}h^2n \cdot \max_{a\le x\le b} f'(x) \\ &= \frac{1}{2}h^2\left(\frac{b-a}{h}\right) \cdot \max_{a\le x\le b} f'(x) \sim O(h) \end{aligned}$$

iii. Composite left point formula: Composite left point numerical integral

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} hf(x_i) + O(h)$$

Actually

$$\int_a^b f(x)\,dx \approx \frac{\sum_{i=0}^{n} y_i}{n}(b-a) := \text{average of } f(x_i) \times \text{interval length}$$

(c) Error analysis midpoint rule:

i. Let $M = \frac{1}{2}(x_{i+1} + x_i)$

$$F(x_{i+1}) = F\Big(M + \frac{h}{2}\Big) = F(M) + \frac{h}{2}F'(M) + \frac{h^2}{8}F''(M) + \frac{h^3}{48}F'''(M) + \dots$$

$$F(x_i) = F\Big(M - \frac{h}{2}\Big) = F(M) - \frac{h}{2}F'(M) + \frac{h^2}{8}F''(M) - \frac{h^3}{48}F'''(M) + \dots$$

Then

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = F(x_{i+1}) - F(x_i)$$

$$= hF'(M) + O(h^3) = hf(M) + \frac{h^3}{24}f''(M) + \dots$$

Error for mid point rule

$$E_i \sim O(h^3)$$

ii. Total error: for some ξ_i in $[M, x_{i+1}]$ and η_i in $[x_{i-1}, M]$

$$E = \sum_{i=1}^{n} \frac{h^3}{48}\Big(f''(\xi_i) + f''(\eta_i)\Big) \le \frac{h^3}{24}\max_{a\le x\le b} f''(x)$$

That is

$$E \sim O(h^2)$$

(d) Trapezoid rule:

i. Recall Taylor series at x_i gives

$$\begin{aligned}
&\int_{x_i}^{x_{i+1}} f(x)\,dx \\
&= hf(x_i) + \frac{h^2}{2}f'(x_i) + \frac{h^3}{6}(f''(x_i)) + \dots \\
&= \frac{1}{2}hf(x_i) + \frac{1}{2}h\Big(f(x_i) + hf'(x_i) + \frac{1}{2}h^2 f''(x_i)\Big) + O(h^3) \\
&= \frac{1}{2}hf(x_i) + \frac{1}{2}hf(x_i + h) + O(h^3) \\
&= \frac{1}{2}[f(x_i) + f(x_i + h)]h + O(h^3)
\end{aligned}$$

Error for trapezoid rule

$$E \sim O(h^2)$$

ii. Composite Trapezoid rule

$$\begin{aligned}\int_a^b f(x)\ dx &= \sum_{i=0}^{n-1}\int_{x_i}^{x_{i+1}} f(x)\ dx\\ &= \sum_{i=0}^{n-1}\Big(\frac{h}{2}[f(x_i)+f(x_i+h)]+O(h^3)\Big)\\ &= \frac{h}{2}\Big(y_0+2y_1+2y_2+\cdots+2y_{n-1}+y_n\Big)+O(h^2)\end{aligned}$$

.3 5.4 Simpson's Rule

1. Simpson's rule:

(a) Idea: Integrate a quadratic on each subinterval.

$$f(x)\approx a_0+a_1x+a_2x^2,\quad x_{i-1}\le x\le x_i$$

Nodes:

$$x_{i-1},\quad \frac{x_{i-1}+x_i}{2},\quad x_i$$

For notation simplicity consider node points

$$a=x_{i-1},\quad a+h=\frac{x_{i-1}+x_i}{2},\quad a+2h=x_i$$

and the integral

$$\int_a^{a+2h} f(x)\ dx = F(a+2h)-F(a)$$

(b) Error analysis: Recall

$$\int_{x_i}^{x_i+h} f(x)\ dx = hf(x_i)+\frac{h^2}{2}f'(x_i)+\frac{h^3}{6}f''(x_i)+\frac{h^4}{24}f'''(x_i)+O(h^5)$$

Then

$$\begin{aligned}\int_a^{a+2h} f(x)\ dx &= 2hf(a)+2h^2f'(a)+\frac{4h^3}{3}f''(a)+\frac{2h^4}{3}f'''(a)+O(h^5)\\ &= h\Big[c_1f(a)+c_2f(a+h)+c_3f(a+2h)\Big]+\dots\end{aligned}$$

Method of undetermined coefficients. Choose c_1, c_2 and c_3 to make it happen!

$$\begin{aligned}&2hf(a)+2h^2f'(a)+\frac{4h^3}{3}f''(a)+\frac{2h^4}{3}f'''(a)+O(h^5)\\ =&h\Big[c_1f(a)+c_2f(a+h)+c_3f(a+2h)\Big]+\dots\\ =&h\Big[c_1f(a)+c_2[f(a)+hf'(a)+\frac{h^2}{2}f''(a)+\frac{h^3}{6}f'''(a)+O(h^4)]\\ &+c_3[f(a)+2hf'(a)+2h^2f''(a)+\frac{4h^3}{3}f'''(a)+O(h^4)]\Big]+\dots\end{aligned}$$

Therefore

$$\begin{cases} c_1+c_2+c_3=2\\ c_2+2c_3=2\\ \dfrac{c_2}{2}+2c_3=\dfrac{4}{3}\end{cases}\Rightarrow\begin{cases} c_1=1/3\\ c_2=4/3\\ c_3=1/3\end{cases}$$

Hence,

$$\int_a^{a+2h} f(x)\ dx = \frac{h}{3}\Big[f(a) + 4f(a+h) + f(a+2h)\Big] + O(h^5)$$

(c) Final result:

Theorem .3 (Replace h by $h/2$)**.** *Simpson's rule with equal spaced node*

$$\int_a^b f(x)\ dx \approx \sum_{i=1}^{n} \frac{h}{6}\Big[f(x_{i-1}) + 4f\Big(\frac{x_{i-1}+x_i}{2}\Big) + f(x_i)\Big]$$

with error

$$E \sim O(h^4)$$

2. Richardson's extrapolation
`https://en.wikipedia.org/wiki/Richardson_extrapolation`

3. Adaptive algorithm idea: Goal is to approximate $\int_a^b f(x)\ dx$ within a given tolerance without knowing the answer to compare with.

(a) Instead of dividing $[a,b]$ evenly and using more parabolas everywhere, check where subdivision is needed. A preset partition: n

- Mid point formula
- Trapezoid rule
- Simpson's rule

Adaptive algorithm:

- Numerical integral: Determine n from tolerance of error
- General: automatically choose parameters according to desired accuracy
- Relate the following

$$e_n \quad \text{and} \quad x_n, \quad x_{n+1}$$

(b) Simpson's rule on interval $[a,b]$

$$S(a,b) = \frac{b-a}{6}\Big[f(a) + 4f\Big(\frac{a+b}{2}\Big) + f(b)\Big]$$

A more careful error

$$E(a,b) = -\frac{1}{90}\Big(\frac{b-a}{2}\Big)^5 f^{(4)}(\xi)$$

for some ξ in $[a,b]$.

(c) Notation:

Partition 1: one interval $[a,b]$ with $h = b-a$

$$\int_a^b f(x)\ dx = S_1 + E_1$$

where

$$S_1 = S(a,b), \quad E_1 = E(a,b) = -\frac{1}{90}\Big(\frac{h}{2}\Big)^5 f^{(4)}(\xi)$$

(d) Partition 2: $[a,c]$, $[c,b]$ where $c = (a+b)/2$

$$\int_a^b f(x)\ dx = S_2 + E_2,$$

where

$$S_2 = S(a,c) + S(c,b)$$

and

$$E_2 = -\frac{1}{90}\Big(\frac{h/2}{2}\Big)^5 f^{(4)}(\xi) - \frac{1}{90}\Big(\frac{h/2}{2}\Big)^5 f^{(4)}(\eta) \approx \frac{1}{16}E_1$$

(e) Subtraction

$$S_2 - S_1 = E_1 - E_2 \approx 15E_2$$

that is

$$E_2 \approx \frac{1}{15}|S_2 - S_1|$$

This will serve as a guideline for adaptive Simpson's rule

(f) Pseudocode

```
function result = asimpson(f,a,b,tol)
h = b-a; c = (a+b)/2; d = (a+c)/2; e = (c+b)/2;
one_simpson = h/6*(f(a)+4*f(c)+f(b));
two_simpson = h/12*(f(a)+4*f(d)+2*f(c)+4*f(e)+f(b));
if(|one_simpson-two_simpson| < tol*15)
    result = one_simpson;
else
    left_simpson = asimpson(f,a,c,tol/2);
    right_simpson = asimpson(f,c,b,tol/2);
    result = left_simpson + right_simpson;
end
```

.4 5.5 Gaussian Quadrature Formulas

1. Quadrature formula

   (a) General idea: find weights w_1, w_2, ..., w_n such that

   $$\int_a^b f(x)\ dx \approx w_0 f(x_0) + w_1 f(x_1) + \cdots + w_n f(x_n)$$

   where the weights are chosen to minimize error.

   - Easy to use
   - Accurate
   - Good for three dimension
   - Used by most numerical algorithms

2. Known sample points:

   (a) Determine the coefficients: Assume we already know the sample points

   $$(x_0, f(x_0)), \quad ..., \quad (x_n, f(x_n))$$

   Lagrange interpolation:

   $$p(x) = \sum_{i=0}^{n} l_i(x) f(x_i), \quad l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

   Then

   $$\int_a^b f(x)\ dx \approx \int_a^b \sum_{i=0}^{n} l_i(x) f(x_i)\ dx = \sum_{i=0}^{n} f(x_i) \int_a^b l_i(x)\ dx$$

Pick

$$w_i = \int_a^b l_i(x)\ dx$$

(b) Example: Find the quadrature formula for

$$\int_0^2 f(x)\ dx \quad \text{with nodes} \quad x_0 = 0, \quad x_1 = 1, \quad x_2 = 2$$

Solution

$$l_0(x) = \frac{1}{2}(x-1)(x-2), \quad l_1(x) = -x(x-2), \quad l_2(x) = \frac{1}{2}x(x-1)$$

and

$$w_0 = \int_0^2 l_0(x)\ dx = \frac{1}{3}, \quad w_1 = \frac{4}{3}, \quad w_2 = \frac{1}{3}$$

Quadrature formula

$$\int_0^2 f(x)\ dx \approx \frac{1}{3}f(0) + \frac{4}{3}f(1) + \frac{1}{3}f(2)$$

Simpson's rule!

3. Unknown sample points:

(a) Quadrature points: What if we don't know the nodes? We get to choose!

- Example.

$$\int_a^b f(x)\ dx \approx c_0 f(x_0) + c_1 f(x_1)$$

Instead, consider

$$\int_{-} 1^1 f(x)\ dx \approx c_0 f(x_0) + c_1 f(x_1)$$

Why is this allowed? Can always do a change of variable which maps $[a,b] \to [-1,1]$. Use the interpolant!

$$\int_a^b f(x)\ dx = \frac{b-a}{2}\int_{-1}^1 f\Big(\frac{b-a}{2}x + \frac{a+b}{2}\Big)\ dx$$

- Pick the nodes smartly to improve accuracy.
- Weights w_i are determined by the interval and the nodes
- w_i are independent of $f(x)$
- Quadrature is reusable for a fixed interval and nodes
- Choosing the nodes for $[-1,1]$
- Gauss – Legendre quadrature points

`https://en.wikipedia.org/wiki/Gaussian_quadrature`

(b) Example:
Example 1: use two points Gaussian Quadrature to approximate

$$\int_{-1}^1 x^2\ dx$$

Example 2: use two points Gaussian Quadrature to approximate

$$\int_0^3 x^2\ dx$$

Example 3: Matlab

(c) Gaussian quadrature: It's amazing!!!
   - Use by most numerical algorithms
   - Easy to use: independent of $f(x)$

     Quadrature table for $[-1,1] \to$ change of interval
   - Very high accuracy!

4. Where do these crazy nodes come from? Orthogonal polynomials are key.

   (a)

   Theorem .4. *Let q be a degree $n+1$ polynomial such that*

$$\int_a^b x^k q(x)\ dx = 0, \quad 0 \le k \le n$$

   Let x_0, $x_1, \dots$, x_n be the zeros of q. Then the formula

$$\int_a^b f(x)\ dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

   will be exact for all polynomials $f(x)$ of degree at most $2n+1$

   q here is an orthogonal polynomial! Gram-Schmidt saves the day again!

   (b) How to interpret?

   When using Gaussian Quadrature formula for intergral
   - 2 nodes $\to n = 1 \to$ order 3 polynomial accuracy
   - 3 nodes $\to n = 2 \to$ order 5 polynomial accuracy
   - 4 nodes $\to n = 3 \to$ order 7 polynomial accuracy

   (c) Proof: First, if $f(x)$ is a degree $\le n$ polynomial, then the formula

$$\int_a^b f(x)\ dx = \sum_{i=0}^{n} w_i f(x_i)$$

   is exact for any $n+1$ nodes by the number of degrees of freedom available. Second, if $f(x)$ is a degree $\le 2n+1$ polynomial, divide f by orthogonal polynomial q

$$f(x) = pq + r$$

   where p, r are degree $\le n$ polynomials Then

$$\int_a^b f(x)\ dx = \int_a^b pq\ dx + \int_a^b r\ dx$$

   By assumption of the theorem

$$\int_a^b pq\ dx = 0$$

   and from case (1) the following formula is exact

$$\int_a^b r\ dx = \sum_{i=0}^{n} w_i r(x_i)$$

   If we choose the nodes to be the zeros of q, then

$$f(x_i) = p(x_i)q(x_i) + r(x_i) = r(x_i)$$

   therefore the formula is exact

$$\int_a^b f(x)\ dx = \int_a^b pq\ dx + \int_a^b r\ dx = 0 + \sum_{i=0}^{n} w_i r(x_i) = \sum_{i=0}^{n} w_i f(x_i)$$

5. Gaussian quadrature: Given interval $[a, b]$, find the gaussian quadrature formula
   - Find the nodes: zeros of q
   - Find w_i and determine the quadrature formula
   - The accuracy is of degree $2n + 1$ polynomial interpolation (the polynomial interpolation for the n+1 is just degree n)
6. Gaussian quadrature formula: Can construct these orthogonal polynomials q via Gram-Schmidt process if you like. Easier approach is to do by hand. Find x_i and w_i on [-1,1].
   - Find the zeros of q where
   $$\int_{-1}^{1} x^k q(x)\, dx = 0, \quad \text{for all } k \leq n$$
   - Example for $n = 1$
   - Homework
7. Runge's phenomenon: Gaussian quadrature formula
   - Matches the error of high order polynomial interpolation.
   - Can handle Runge's phenomenon!!!
8. Gaussian quadrature: Gaussian quadrature formula is very popular
   - Well established
   - Accurate
   - Efficient

   Many more can be said about Gaussian quadrature
   - 3-D integration
   - Integral with infinity on one side
   $$\int_{2}^{\infty} f(x)\, dx$$

Chapter 8 Ordinary Differential Equations

.1 8.1 Initial value problem and Taylor series method

1. Differential equation:
   (a) Motivation: Consider a closed region
      - Time: t
      - Population: $y(t)$
      - Birth rate: 20%
      - Death rate: 10%
      - Assume unlimited resources
      - No one moves in or out

      Questions: What's a function for the population with respect to t?

(b) Initial value problem: The population satisfies

$$y' = 0.1y$$

Solution

$$y = Ce^{0.1t}$$

Need an initial condition for uniqueness: Say $y(0) = C_0 = 2000$, plug in

$$C = 2000$$

Finally the population function

$$y = 2000e^{0.1t}$$

2. Initial value problem:

   (a) Ordinary differential equation (ODE)
   - an equation that involves derivative
   - The order of ODE: the highest order of derivative

   (b) Initial value problem (IVP)
   - ODE plus
   - Initial condition

3. Example: Second order ODE

$$y'' = y^2(1 - y')$$

Second order IVP

$$\begin{cases} y'' = y^2(1 - y') \\ y(0) = 1 \\ y'(0) = 4 \end{cases}$$

4. General setup:

   (a) General setup of differential equations.

$$y' = f(t, y), a \leq t \leq b, \quad y(0) = y_0$$
$$y'' = f(t, y, y'), a \leq t \leq b, \quad y(0) = y_0, y'(0) = y_0'$$

   (b) How can we ensure we have a solution?

   Theorem .1 (Existence and uniqueness theorem). *An IVP has a unique solution only if number of initial condition equal the order of the ODE*

   (c) Comments:
   - IVP is very useful
   - solving IVP analytically is very difficult
   - In general, numerical solution of IVP is needed

5. Numerical solution of IVP:

   (a) Numerical solution of the first order IVP

$$\begin{cases} y' = 0.1y \\ y(0) = 2000 \end{cases}$$

   Consider time step h and $t_i = i * h$. The numerical derivatives

$$y'(t_i) \approx \delta y_i = \frac{y(t_{i+1}) - y(t_i)}{h}$$

   Replace $y'(t_i)$ by $0.1y(t_i)$ then the ODE implies

$$\delta y_i = 0.1y(t_i) \quad \rightarrow \quad y(t_{i+1}) = (1 + 0.1h)y(t_i)$$

(b) Numerical solution of IVP: Finite difference method
    i. Define a time step h
    ii. Replace the derivatives in ODE by numerical derivative
    iii. Define an iterative scheme
    iv. Start from initial value to generate values on all the nodes
    v. Let $h \to 0$ to reduce the error
    vi. Issue is that error accumulates with each iteration.

6. Euler's method: order 1 Taylor series method:
   Consider first order IVP on time interval [a,b]

$$\begin{cases} y' = f(t,y) \\ y(0) = y_0 \end{cases}$$

   (a) Partition with n intervals
   (b) Time step: $h = (b-a)/n$
   (c) Let $t_i = i * h$ and y_i be the approximation of $y(t_i)$
   (d) Numerical scheme (forward Euler method)

$$y_{i+1} = y_i + hf(t_i, y_i)$$

   (e) Equates to following the tangent line at each step. Draw picture to illustrate.

7. Error estimate: Sample error estimate for

$$y' = 0.1y$$

   Taylor series

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}f''(\xi) = y(t_i) + h0.1y(t_i) + \frac{h^2}{2}y''(\xi_i)$$

   Scheme

$$\hat{y}_{i+1} = \hat{y}_i + 0.1h\hat{y}_i$$

   Subtraction

$$e_{i+1} = (1+0.1h)e_i + \frac{h^2}{2}y''(\xi_i) \leq ke_i + Mh^2$$

   where

$$k = 1 + 0.1h, \quad M = \max_{a \leq x \leq b} \frac{y''(x)}{2}$$

8. Error analysis i = 0

$$e_1 \leq ke_0 + Mh^2 = Mh^2$$

   i = 1

$$e_2 \leq ke_1 + Mh^2 \leq k(Mh^2) + Mh^2 = (1+k)Mh^2$$

   i = 2

$$e_3 \leq (ke_2 + Mh^2) \leq (1 + k + k^2)Mh^2$$

   Recursively

$$e_n \leq (1 + k + \cdots + k^{n-1})Mh^2$$

9. Error analysis: Geometric sequence

$$1 + k + \cdots + k^{n-1} = \frac{1-k^n}{1-k}$$

therefore (recall $k = 1 + 0.1h$)

$$e_n \leq \frac{k^n - 1}{k-1} M h^2 = 10Mh((1+0.1h)^n - 1)$$

Finally

$$e_n \sim Ch \cdot \Big(e^{b-a} - 1\Big)$$

where C is some constant independent of h

10. Error analysis: Error analysis for Euler's method

$$e_n \sim Ch \cdot \Big(e^{b-a} - 1\Big)$$

Accumulated error

- Local error: $O(h^2)$
- Global error: if $[a, b]$ is a bounded interval, $e_i \sim O(h)$
- Global error increases as t_i moves away from a
- The error blows up for unbounded interval
- The error estimate requires work

11. Error analysis

Theorem .2 (Global error estimate for Euler's method). *Consdier the IVP*

$$\begin{cases} y' = f(t, y) \\ y(0) = y_0 \end{cases}$$

If there exist constant L, M such that

$$\left|\frac{\partial f}{\partial y}\right| \leq L, \quad y'' \leq M$$

then

$$|y(t_i) - y_i| \leq \frac{hM}{2L}\Big(e^{L(t_i - a)} - 1\Big), \quad 0 \leq i \leq n$$

12. Example: Solve IVP

$$\begin{cases} y' = 0.1y(1 - \frac{y}{500}) \\ y(0) = 20 \end{cases}$$

- Logistic equation $y' = ky(1 - \dfrac{y}{R})$
  - y: population
  - k: growth rate
  - R: maximal capacity
- Logistic (Sigmoid) function

$$y = \frac{1}{1 + e^{-x}}$$

13. Stability

- Stability of the system
  - Accumulated error
  - Butterfly effect
  - Chaos
  - Stable/unstable
- Stabiltiy of the numerical scheme

.2 8.2 More on ODE

1. Higher order of convergence: Forward Euler method
$$\delta y_i = \frac{y_{i+1} - y_i}{h}$$
Backward Euler method
$$\delta y_i = \frac{y_i - y_{i-1}}{h}$$
Central difference
$$\delta y_i = \frac{y_{i+1} - y_{i-1}}{2h}$$
2. Higher order of convergence: Example
$$\begin{cases} y' = -3y \\ y(0) = 1 \end{cases}$$
Forward Euler method
$$y_{i+1} = (1 - 3h)y_i$$
Backward Euler method
$$y_{i+1} = \frac{y_i}{1 + 3h}$$
Central difference
$$y_{i+1} = -6hy_i + y_{i-1}$$
Solving is much harder for nonlinear f on RHS.
3. Comparison
   - Forward Euler: explicit and easy.
   - Backward Euler: implicit and stabler.
   - Central difference: not necessarily better.
   - Need to improve accuracy in a different way.
   - Accumulated error.
4. Stability: Example
$$\begin{cases} y' = -3y \\ y(0) = 1 \end{cases}$$
Forward Euler method
$$y_{i+1} = (1 - 3h)y_i$$
Backward Euler method
$$y_{i+1} = \frac{y_i}{1 + 3h}$$
Consider interval $[0, 10]$ with different h.

5. Taylor series method: Improve the accuracy of Taylor series. Consider

$$\begin{cases} y' = y(1-y) \\ y(0) = 1/2 \end{cases}, \quad \text{Exact solution: } y = \frac{1}{1+e^{-x}}$$

Taylor series for higher order

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + \frac{h^3}{6}y'''(t) + \dots$$

Take the derivative to find the value we need

$$\begin{aligned} y' &= y(1-y) \\ y'' &= y'(1-y) + y(-y') = y' - 2yy' \\ y''' &= y'' - 2y'y' - 2yy'' \end{aligned}$$

6. Taylor series method
   - Higher derivative gives higher order
   - Compute high order derivative via calculus
   - Balance between computational cost and accuracy

7. Quadrature method: Consider

$$y' = f(t,y)$$

Quadrature method

$$y(t_{i+1}) - y(t_i) = \int_{t_i}^{t_{i+1}} y'(s)\,ds = \int_{t_i}^{t_{i+1}} f(s, y(s))\,ds$$

then use quadrature formula to approximate the integral

8. Higher order ODE: Consider

$$\begin{cases} y'' = -y \\ y(0) = 0 \\ y'(0) = 1 \end{cases}, \quad \text{Exact solution: } y = \sin(t)$$

Numerical derivative

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

then

$$y_{i+1} = (2-h^2)y_i - y_{i-1}, \quad y_0 = 0,$$

and

$$y'(0) = 1 \to \frac{y_1 - y_0}{h} = 1 \to y_1 = h$$

9. ODE system: Predator and prey model

$$\begin{aligned} &W' = -0.1(1 - \tfrac{S}{200})W \\ &S' = 0.3(1 - \tfrac{W}{500})S \end{aligned}$$

   - W: wolf population
   - S: sheep population
   - 200: enough sheep
   - 500: too many wolves

Last day: Career Advice

What follows are my views on careers based on my experience. Take or leave it. There are 2 main paths one treads after graduation....

1. Graduate school
   - (a) What is your end goal?
     - Academia teaching
     - Academia research
     - Academia mix
     - Top industry job
     - Goverment lab / R and D
   - (b) How to get there? Show capable of next level
     - Specialize: take above requirements in upper level
     - Top scores in key classes: Abstract algebra and Real analysis for anything math
     - GRE score sometimes matters
   - (c) How to set yourself apart? 200+ applicants for 15 spots.
     - Research: Indep research (REU best, indep study, capstone project)
     - Accomplish: Papers, talks, conferences
     - Connections: At grad school, local profs
   - (d) What skills do you need?
     - Interest
     - Work hard
     - Resourcefulness
     - Resiliance (they will try to destroy you)
     - Strategy: Pick a path and specialize from start (research, teaching, industry)
     - Smartest aren't best in my experience
   - (e) Proof: `https://www.math.wisc.edu/graduate`
2. Job (anything tech related)
   - (a) Top skills
     - Specific skills: Coding, data, math reasoning
     - Broad skills: Attention to detail, communication, ability to abstract
     - Go to a career fair and ask!
   - (b) The trifecta: MATH + CS + STAT (in that order)
     - What if just MATH? You are smart but you can't do anything. No integrals to do anymore...
     - What if just CS? You will be the IT coder, website design, database, etc. No chance to move.
     - What if just STAT? You will run stats all day and generate reports. The end.
     - What if all three: Own large projects, innovate, problem solve, manage a team, leadership. Talk about Century Link and building large systems to flag network outages. Only 5 guys needed.
   - (c) Job titles: What do you think they do?
     - Business analyst (sit between business and tech)
     - Data analyst (code, data grunt work and small projects, upalumpa ish)
     - Data scientist (don't code much, manage teams and entire projects)

(d) How to get there?

   - Build skills specific to job ads. No class, find on online. If credible, put on resume.
   - Know your skills and limitations, be able to communicate them. Why are you better?
   - Connections: Meet people at job fair. Shake hands, get cards, send email thanking and asking for info. They will remember.
   - Get experience: Internship, PIC math, indep study, online, volunteer, innovate.

3. Proof:

   - Job ads: `http://www.mayoclinic.org/jobs`, `https://www.myinterfase.com/uwlax/employer/`
   - Pay: `http://www.payscale.com/`