Numerical Analysis

Nadav Dym

March 15, 2023

Warning: Unlike textbooks, these notes have no serious proof-reading and certainly contain some errors. If you spot such an error, please let me know vie email. Using notes as a replacement for lecture attendance is not recommended.

Lesson 1

1 Introduction

This course addresses the following questions: of all the math we learned so far- what can we compute with a computer, and how? In school we learn how to solve by hands equations like

$$12 \times 7 =?$$
$$x^2 + 8x - 1 = 0$$
$$\int_0^1 x^5 dx =?$$

or equations in two variables like

$$x + y = 10$$
$$2x - y = 3$$
$$x, y =?$$

Some math questions we know how to do, but we'd prefer to do with a calculator. For example

$$1,985,637 \times 9,936,464 =?$$

most humans would try to avoid this type of calculations. Computers can do thousands of these calculations in less than a second, without making any error (well, we'll see about that). Essentially, this is all computers give us: the ability to make calculations which we know how to do, fast and efficiently. So asking: how would you do this calculation with a computer? Is like asking: how would you do this calculations if you had a lot of paper, pens, time, and patience...

Let's now see some examples from our university educations which we know a lot about. Can you suggest a method to solve these problems?

Numerical analysis

$$\sin(0.67) =?$$
$$\int_0^1 e^{-x^2} dx$$
$$x^2 - sin(x) - 1 = 0$$

Numerical linear algebra

$$Ax = b \text{ (we assume } A \in \mathbb{R}^{n \times n}, det(A) \neq 0)$$
$$\text{what are the eigenvalues of } A? \text{ (assume } A = A^T)$$
$$det(A) =?$$

When we have different methods to solve a problem, we compare the methods through how fast they can solve the problem, which we can count through the number of basic arithmetic operations we need to solve the problem: For example, the determinant of $A = (a_{ij})_{1\leq i,j\leq n}$ from above can be written as

$$\det(A) = \sum_{\sigma\in S_n} sign(\sigma) a_{1\sigma(1)} \cdot a_{2\sigma(2)} \cdot \ldots \cdot a_{n\sigma(n)}$$

For each permutation, we have n products, and so altogether we have $n(n!)$ product operations and $n!-1$ summation operations, for a total of $n(n!) + n! - 1$ operations. An alternative algorithm would be to use eigendecomposition: assume we can compute an eigendecomposition $A = U^T DU$ of A where U is orthogonal and D is a diagonal matrix, and this takes us Cn^3 operations. Since

$$\det(A) = \det(D) = D_{11} \cdot D_{22} \cdot \ldots \cdot D_{nn}$$

we can compute $\det(A)$ in Cn^3+n-1 operations. So this algorithm is much faster (in practice other decompositions of A are usually used).

1.1 Binary and base b expansions

We usually use base-10 expansions to represent numbers. That is $x = 453.62$ means

$$x = 4 \times 10^2 + 5 \times 10 + 3 \times 10^0 + 6 \times 10^{-1} + 2 \times 10^{-2}.$$

The number x above has a finite base-10 expansion. All non-negative real numbers have a possibly infinite base-10 expansion: there exists $L \in \mathbb{Z}$ and a sequence $(c_\ell)_{\ell=L}^{\infty}$ where each c_ℓ is in $\{0, 1, \ldots, 9\}$, such that

$$x = \sum_{\ell=L}^{\infty} c_\ell 10^{-\ell}.$$

Similarly, for any integer $b \geq 2$, every positive integer n has a finite expansion

$$n = \sum_{\ell=0}^{L} a_\ell b^\ell, a_0, \ldots, a_L \in \{0, 1, \ldots, b-1\}$$

Question 1. *What is the base 2 expansion of $n = 19$?*

Answer 1. *We do this by recursively finding k such that $2^k \leq n < 2^{k+1}$, defining a new number $n - 2^k$ which by definition satisfies*

$$0 \leq n - 2^k < 2^k,$$

and then reiterating...

Note that

$$2^4 < 19 < 2^5$$

and

$$2 < 19 - 2^4 = 3 < 2^2$$

and

$$0 < 19 - 2^4 - 2 = 1 = 2^0$$

and so altogether, Imitating our usual base 10 notation, we can rewrite this as

$$(10011)_2 = 2^4 + 2^1 + 2^0 = 19.$$

Question 2. ***Question*** *What is the number $(102.2)_3$?*

Answer 2.

$$(102.2)_3 = 2 * 3^{-1} + 2 * 3^0 + 1 * 3^2 = 11 + 2/3$$

Any real positive number has a (finite or infinite) expansion in base b, for any natural $b \geq 2$. We prove this first for x in $[0, 1)$:

Theorem 1. *Let x be a real number in $[0,1)$, and $b \geq 2$ an integer. Then there exists a sequence $(c_\ell)_{\ell=1}^{\infty}$ where each c_ℓ is in $\{0,1,\ldots,b-1\}$, such that*

$$x = \sum_{\ell=1}^{\infty} c_\ell b^{-\ell}.$$

Why is this relevant? Computers like working in base 2, since deep inside everything is based on $0-1$ gates, while we like working in base 10. This theorem shows that both bases are equally legitimate.

Proof. We first prove the following simple lemma

Lemma 1. *If $b \geq 2$ is an integer, $L \in \mathbb{N}$ and $x \in [0, b^{-L+1})$, then there exists $c_L \in \{0,1,\ldots,b-1\}$ such that $x - c_L b^{-L} \in [0, b^{-L})$*

Proof. we note that

$$\cup_{j=0}^{b-1}[jb^{-L}, (j+1)b^{-L}) = [0, b^{-L+1}).$$

So there exists $c_L \in \{0,1,\ldots,b-1\}$ so that $x \in [c_L b^{-L}, (c_L+1)b^{-L})$. Then

$$r_L = x - c_L b^{-L} \in [0, b^{-L})$$

□

We now prove the theorem. By assumption $x \in [0, b^0)$. We now define $c_1, c_2, c_3, \ldots$ recursively as follows: by Lemma 1 with $L = -1$ there exists c_1 such that

$$r_1 = x - c_1 b^{-1} \in [0, b^{-1}).$$

Next, we see by Lemma 1 that there exists $c_2 \in \{0,1,\ldots,b-1\}$ such that

$$r_2 = x - (c_2 b^{-2} + c_1 b^{-1}) = r_1 - c_1 b^{-1} \in [0, b^{-2})$$

continuing in this way we get a sequence $(c_\ell)_{\ell=1}^{\infty}$ and $(r_\ell)_{\ell=1}^{\infty}$ such that for every $S \in \mathbb{N}$,

$$r_S = x - \sum_{\ell=1}^{S} c_\ell b^{-\ell} \in [0, b^{-S}).$$

This implies that $r_S \to 0$ as $S \to \infty$, and by definition of converging sums this means that

$$x = \sum_{\ell=1}^{\infty} c_\ell b^{-\ell}.$$

□

Conclusion We deduce from the theorem that for $x > 0$, we can find L large enough so that $x < b^L$ and so xb^{-L} is in $[0,1)$ and we can write

$$xb^{-L} = \sum_{\ell=1}^{\infty} c_\ell b^{-\ell}$$

and so

$$x = b^L \sum_{\ell=1}^{\infty} c_\ell b^{-\ell} = c_1 b^{L-1} + c_2 b^{L-2} + \ldots$$

Question 3. *Write the following numbers as rational numbers*

1. $(1.111\ldots)_2$
2. $(0.10101010\ldots)_2$

Answer 3. *For the first question we have by definition*

$$(1.111\ldots)_2 = 2^0 + 2^{-1} + 2^{-2} + \ldots = \sum_{k=0}^{\infty} (1/2)^k = 2$$

Denoting the second number by x, we see that

$$4x - x = (10)_2 = 2$$

so $x = 2/3$.

1.2 Representing numbers on computers

In numerical calculations we alot a fixed amount of memory for each number. In single precision we alot 32 bits, and in double precision we alot 64 bits. A bit is a digit which is either 0 or 1.

Integers We can represent *integers* using a single bit for the sign, and 31 bits encoding the binary expansion of the number. So a sequence $(c_0, \ldots, c_{31})$ with each c_j in $\{0, 1\}\}$, represents the integer n given by

$$n = (-1)^{c_{31}} \sum_{j=0}^{30} c_j 2^j = (-1)^{c_{31}} (c_3 0, \ldots c_1, c_0)_2$$

In this way we can represent all integers in $[-(2^{31} - 1), 2^{31} - 1]$. If for example we add together two integers and get an integer larger that 2^{31}, the computer will give us the result Inf. Note that this method is not necessarily optimal, or what is used in practice. For example, the number zero is represent twice...

Fixed point representation (not typically used) How to represent rational numbers with 32 bits? One possibility would be to alot the last bit for the sign, and write numbers as

$$x = (-1)^{c_{31}} (c_3 0, \ldots c_{15}.c_{14}, \ldots c_1, c_0)_2 = (-1)^{c_{31}} c_{30} 2^{15} + c_{29} 2^{14} + \ldots + c_{15} + \ldots + c_0 2^{-15}$$

Question 4. *What is the largest number which can be written in this form*

Answer 4.

$$x = 2^{15} + 2^{14} + \ldots + 2^{-15} = 2^{-15} \left(2^{30} + \ldots + 1\right) = 2^{-15} \left(2^{31} - 1\right)$$

Lesson 2

Floating point representations (what we will use) Floating point representations allow for larger (and smaller) numbers. Note that any **non-zero** real number can be written as

$$x = (-1)^s [1.f]_2 \times 2^m \tag{1}$$

where f can have an infinite number of $0-1$ digits and m as an integer. for example

$$3/8 = 1/4 + 1/8 = (0.011)_2 = (1.1)_2 \times 2^{-2}$$

In the floating point representation we use (3) but we limit f to have a finite number of digits, and m to be a finite collection of integers. In single precision, we use a single bit to encode the sign or s, eight bits to encode m, and the remaining 23 bits to encode f. With 8 bits, we can encode $2^8 = 256$ integers. We use the first and last integer to encode $0, -0, \infty, -\infty, NaN$, and then m can assume 254 values: $-126, -125, \ldots, 126, 127$. In more detail, we use the eight bits to define a non-negative integer $e = (e_7, \ldots, e_0)_2$ between 0 and 255, we keep the values $e = 0, e = 255$ to denote special values, and define

$$m = e - 127.$$

The largest floating point number is

$$2^{127} \leq x = (1.1, 1 \ldots 1) \times 2^{127} \leq 2^{128}$$

far larger than what we got using fixed point, or even when considering only integers.

We call $[1.f]_2$ the mantissa, f the normalized mantissa, m the exponent, and e the biased exponent.

Double precision In double precision (which is the standard in Matlab) we use 64 bits to store each number: one bit for , 11 bits for the exponent, and 52 bits for the normalized mantissa. We focus on single precision in our discussion unless mentioned otherwise.

Question 5. *What is the closest number to* 1 *from above and below, in single and double precision?*

Answer 5. *In single precision the closest number from above is*

$$(1.00 \ldots 01)_2 = 1 + 2^{-23}$$

and the closest number from below is

$$(1.11 \ldots 1) \times 2^{-1} = \sum_{k=1}^{24} 2^{-k} = 1 - 2^{-24}$$

Denoting $\epsilon_{\text{single}} = 2^{-23}$*, the closest number to one from above is* $1 + \epsilon_{\text{single}}$ *and the closest number from below is* $1 - \epsilon_{\text{single}}/2$*. In double precision the closest numbers to one from above and below are* $1 + \epsilon_{\text{double}}$ *and* $1 - \epsilon_{\text{double}}/2$*, where* $\epsilon_{\text{double}} = 2^{-52}$.

Rounding and error measurements Lets call number which can be exactly represented in single precision *computer numbers*. If we apply some arithmetic operation (multiplication/division/addition/subtraction) to two computer numbers, the number of digits in the mantissa may increase, and as a result the new number may not be a computer number. If we get a non-zero number of the form

$$x = (-1)^s [1.f]_2 \times 2^m$$

where $-126 \leq m \leq 127$and f has more than 23 digits, we round according to the 24-th digit: If it is 1 we add one to the 23rd digit and remove all digits 24 and up. Otherwise we remove 24-th digit and up without changing the 23rd digit. We denote the resulting number by $fl(x)$. What is the *absolute error* $|fl(x) - x|$ of this procedure?

We can write

$$x = (-1)^s (1.f)_2 \times 2^m = (-1)^s \left(1 + \sum_{k=1}^{\infty} f_k 2^{-k}\right) \times 2^m$$

where $f_k \in \{0, 1\}$. We then have

$$|fl(x) - x| = \left(1 + \sum_{k=25}^{\infty} f_k 2^{-k}\right) \times 2^m \leq 2^m 2^{-24}$$

This error can be very large. However a more natural error estimate is the relative error

$$|fl(x) - x|/|x| \leq \frac{2^m 2^{-24}}{[1.f]_2 \times 2^m} \leq \frac{2^m 2^{-24}}{2^m} \leq 2^{-24}$$

so the relative error of rounding is pretty low. In double precision where we have 52 bits for the normalized mantissa the relative error is bounded by $2^{-53} \sim 2 \times 10^{-16}$. Note that the rounding error is $\frac{\epsilon_{\text{single}}}{2}$ for single precision and $\frac{\epsilon_{\text{double}}}{2}$ for double precision.

Remark: If $f(x)$ is some approximation of $x \neq 0$, the relative error

$$\frac{|f(x) - x|}{|x|} = \epsilon$$

if and only if

$$f(x) = x(1 + \delta) \tag{2}$$

and $|\delta| = \epsilon$. This is because (2) implies that

$$\frac{f(x) - x}{x} = \delta$$

Error accumulation When we start doing calculations, errors build up gradually. Let us consider what happens when we multiply, divide, sum or take difference of two positive numbers x_1, x_2, which themselves are rounded:

Let us assume that $x_1, x_2 > 0$, that y_1, y_2 are two number such that

$$y_1 = (1 + \delta_1)x_1 \text{ and } y_2 = (1 + \delta_2)x_2,$$

and

$$|\delta_1|, |\delta_2| \leq \epsilon \leq 1/2.$$

Now let us consider the approximate error of approximating addition, subtraction, multiplication and division of x_1, x_2 by the same operations on y_1, y_2

Addition For addition we have

$$y_1 + y_2 = (1 + \delta_1)x_1 + (1 + \delta_2)x_2 = \left[1 + \frac{\delta_1 x_1 + \delta_2 x_2}{x_1 + x_2}\right]$$

and since

$$\left|\frac{\delta_1 x_1 + \delta_2 x_2}{x_1 + x_2}\right| \leq |\delta_1| + |\delta_2| \leq 2\epsilon$$

we see that the error of addition is up to twice as large as the approximation error or each number separately.

Multiplication

$$y_1 y_2 = (1 + \delta_1)(1 + \delta_2)x_1 x_2 = \left[1 + (\delta_1 + \delta_2 + \delta_1\delta_2)\right] x_1 x_2$$

and since

$$|\delta_1 + \delta_2 + \delta_1\delta_2| \leq 2\epsilon + \epsilon^2$$

we see that the relative error of multiplication of two numbers is also not much worse than the approximation error or each number separately.

Division

$$\frac{y_1}{y_2} = \frac{1 + \delta_1}{1 + \delta_2}\frac{x_1}{x_2} = \left(\frac{1 + \delta_2}{1 + \delta_2} + \frac{\delta_1 - \delta_2}{1 + \delta_2}\right)\frac{x_1}{x_2}$$

and since

$$\left|\frac{\delta_1 - \delta_2}{1 + \delta_2}\right| \leq \frac{2\epsilon}{0.5} = 4\epsilon$$

we see that the relative error of division of two numbers is also not much worse than the approximation error or each number separately.

Subtraction For subtraction we have issues... For example in single precision, where the normalized mantissa f has 23 digits, consider the numbers

$$x_1 = 1 + 2^{-25} \text{ and } x_2 = 1$$

we get $y_1 := fl(x_1) = 1$ and $y_2 := fl(x_2) = 1$. We saw that the relative error of this computation is small, namely 2^{-24}. However since $y_1 - y_2 = 0$, the relative error of estimating $x_2 - x_1$ by $y_2 - y_1$ is

$$\frac{|x_2 - x_1| - y_2 - y_1}{|x_2 - x_1|} = 1.$$

This error occurs when $x_2 - x_1$ is very small compared to x_2.

Question 6. *Find a pair of points $x_1, x_2 > 0$ such that the relative error $\frac{|fl(x_1)-fl(x_2)-(x_1-x_2)|}{|x_1-x_2|} > 10^8$?*

Finally, we show that when $|x_2 - x_1|$ is not small, we can get a stable appoximation for differences.

Theorem 2. *Assume that $x_2 > x_1 > 0$ and $\frac{x_2}{x_2-x_1} < a$ for some a. Assume that y_1, y_2 satisfy*

$$y_1 = (1 + \delta_1)x_1, y_2 = (1 + \delta_2)x_2$$

where $|\delta_1|, |\delta_2| < \epsilon$. Then

$$y_2 - y_1 = (1 + \delta)(x_2 - x_1)$$

where $|\delta| < \epsilon + 2a\epsilon$.

Proof.

$$\begin{aligned} y_2 - y_1 &= (1 + \delta_2)x_2 - (1 + \delta_1)x_1 \\ &= x_2 - x_1 + \delta_1(x_2 - x_1) + (\delta_2 - \delta_1)(x_2 - x_1)\frac{x_2}{x_2 - x_1} \\ &= (x_2 - x_1)\left[1 + (\delta_1 + (\delta_2 - \delta_1)\frac{x_2}{x_2 - x_1})\right] \end{aligned}$$

where

$$|\delta_1 + (\delta_2 - \delta_1)\frac{x_2}{x_2 - x_1}| \leq \epsilon + 2a\epsilon$$

□

Tricks **First example:** Computing

$$f(x) = x - \sin(x)$$

directly for x close to zero, is less accurate than truncating the Taylor expansion of $\sin(x)$ to obtain

$$f(x) = -\frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots$$

Note that while it is true that $\frac{x^3}{3!}$ and $\frac{x^5}{5!}$ are both small when x is small,

$$\lim_{x \to 0} \frac{\frac{x^3}{3!}}{\frac{x^3}{3!} - \frac{x^5}{5!}} = 1,$$

and so by Theorem 2 we can see that the error of computing $\frac{x^3}{3!} - \frac{x^5}{5!}$ will not be very small. In contrast for positive x close enough to zero, we have $x > \sin(x) > 0$ and

$$\lim_{x \to 0} \frac{x}{x - \sin(x)} = \infty$$

Second example: Computing

$$\sqrt{x^2 + 1} - 1$$

is less accurate than computing

$$(\sqrt{x^2 + 1} - 1)\frac{\sqrt{x^2 + 1} + 1}{\sqrt{x^2 + 1} + 1} = \frac{x^2}{\sqrt{x^2 + 1} + 1}$$

Lesson 3

Overflow and Underflow Recall that single precision number are written as

$$x = (-1)^s(1.f_1, \ldots, f_{23})_2 \times 2^m, \tag{3}$$

and the possible values for m are all integers between -126 and 127. It follows that we cannot represent numbers larger than 2^{128}. If our calculations bring us to such a number, then it will be replaced in the computer by the value ∞. This is called overflow. Similarly a number smaller than -2^{128} will be represented by $-Inf$, and a number x with $|x| < 2^{-126}$ will be replace with zero. This is called underflow.

Softmax and avoiding overflow The softmax functions is an interesting example where overflow and underflow may occur. For a fixed $\alpha \geq 0$, the softmax function $s^\alpha : \mathbb{R}^d \to \mathbb{R}^d$ is defined by

$$s_i^\alpha(\vec{x}) = \frac{e^{\alpha x_i}}{\sum_{j=1}^d e^{\alpha x_j}}$$

Here are some properties of these function:

Proposition 1. *1. For all $\vec{x} \in \mathbb{R}^d$ and $\alpha \geq 0$, we have that the entries of $s^\alpha(\vec{x})$ are all positive and sum to one.*

2. If $\alpha = 0$ then $s^0(\vec{x}) = [1/d, 1/d, \ldots, 1/d]$ for all $\vec{x} \in \mathbb{R}^d$.

3. If $x_i \geq x_j$ then $s_i^\alpha(\vec{x}) \geq s_j^\alpha(\vec{x})$.

4. If k is some index such that $x_k > x_i$ for all $i \neq k$, then

$$\lim_{\alpha \to \infty} s^\alpha(\vec{x}) = e_k.$$

where e_k is the unit vector with 1 in the k-th entry and zero otherwise. Thus for large α softmax gives a strong idication of where the maximum of the vector is located.

Proof. Follows easily from the definitions. □

Softmax is used in machine learning for **Classification tasks**. For example, assume you have a collection of family photographs. The family has d different family members and you assign some order to them. We want to build a function h whose input is some large vector X representing a family photograph, and the output is d numbers $(h_1(X), \ldots, h_d(X))$, where $h_i(X)$ represents the probabiliy of the photograph X depicting the i-th family member. Naturally, we would like to build the function h so that $h(X)$ is a probability vector, i.e., it is non-negative and its coordinates sum to one. The standard way of doing this is writing $h(X) = s^\alpha \hat{h}(X)$, where $\hat{h}$ is any function whose image is in $\mathbb{R}^d$. Of course what is really challenging is to get h to give the correct probability vector for each given X, but how this is done is outside our scope right now.

In out context what is interesting about this function is that although its values are between 0 and 1, the exponentiation can lead to overflow and underflow. Overflow will lead to complete nonsense, while underflow just leads to very small terms being rounded to zero. To avoid overflow when computing softmax, we observe that if M is some scalar, and we denote by 1_d the d-dimensional vectors whose entries are all 1, then we have that $s^\alpha(\vec{x}) = s^\alpha(\vec{x} - M1_d)$. Accordingly, if we choose M to be the largest value of $\vec{x}$, we can compute $s^\alpha(\vec{x} - M1_d)$ which does not exponentiate positive numbers and hence will not suffer from overflow, and obtain $s^\alpha(\vec{x})$.

Examples In class we will see some examples in matlab illustrating what we studied until now. This code will be made available on Moodle.

1.3 Alternatives to numerical computations

Symbolic computations An alternative to numerical calculations is symbolic computations. This is used by Mathematica, Maple, Wolfram Alpha and can also be done in Matlab though this is not the default behavior. This means:

1. When multiplying numbers, the number of digits increases and there is no truncation.
2. solutions of equations like $x^2 - 2 = 0$ are given symbolically $x_+ = \sqrt{2}, x_- = -\sqrt{2}$.

The advantage is accuracy, disadvantage is speed...

Interval arithmetic Interval arithmetic is an intermediate choice between symbolic and numerical calculations. The idea is that since our calculations are not exact we represent each number by an interval which we are certain contains it. For example, say we are given some number x which has 'too many digits' and its computer rounding is $\hat{x}$, and the (absolute) error of rounding is bounded by ϵ. We can then represent x by an interval $I_x = [\hat{x} - \epsilon, \hat{x} + \epsilon]$. We know for sure that $x \in I_x$. Say we are given $I_x = [a_x, b_x]$ and $I_y = [a_y, b_y]$. We then define operations such as $I_x + I_y = [a_x + a_y, b_x + b_y]$, so that we are certain that $x + y \in I_x + I_y$

Question: How would you define $I_x \times I_y$ so that if $x \in I_x, y \in I_y$ we get $xy \in I_x \times I_y$? Here are some examples

1. If $a_x, a_y \geq 0$ then we can define $I_x \times I_y = [a_x a_y, b_x, b_y]$.
2. If $b_x, b_y \leq 0$ then we can define $I_x \times I_y = [b_x b_y, a_x a_y]$.
3. If $a_x \leq 0 \leq b_x$ and $a_y \geq 0$: the smallest xy could be is the negative number $a_x b_y$, and the largest it could be is $b_x b_y$ so we get
$$I_x \times I_y = [a_x b_y, b_x b_y]$$
4. If $a_y \leq 0 \leq b_y$ and $a_x \geq 0$ similar idea.
5. If $a_y \leq 0 \leq b_y$ and $a_x \leq 0 \leq b_x$: xy could be negative if $x \in [a_x, 0]$ and $y \in [0, b_y]$ or $x \in [0, b_x]$ and $y \in [a_y, 0]$. So the smallest xy could be is $\min a_x b_y, a_y b_x$. If $x \in [0, b_x]$ and $y \in [0, b_y]$ or $x \in [a_x, 0], y \in [a_y, 0]$ we get $xy \geq 0$: the maximal value xy could be is $\max\{a_x a_y, b_x b_y\}$.

altoger there there are nine different sign arrangements. Instead of checking all of them, we can prove the following simple proposition

Proposition 2. *The minimum and maximum of the function $f(x, y) = xy$ over the rectangle $I_x \times I_y = [a_x, b_x] \times [a_y, b_y]$ are obtained at the corners.*

Based on the proposition, we can define $I_x \times I_y$ by computing the value of xy on the four corners, and taking the edges of the interval to be the smallest and largest values.

Proof. Targil. □

Question 7. *How to define* $\sin(I_x)$*?*

As before, the advantage of interval arithmetic is the ability to keep track of worst case errors. Disadvantages are time, and the fact that as calculations progress worst case errors suggest a very pessimistic picture which is not so realistic.

Lesson 4

1.4 Big-$\mathcal{O}$ notation

An algorithm for a given problem should be able to solve the problem, and do this as efficiently as possible. To discuss the efficiency of algorithms it is convenient to use Big $\mathcal{O}$ notation:

Definition 1. *Let $(x_n)_{n\in\mathbb{N}}$ and $(\alpha_n)_{n\in\mathbb{N}}$. We say that $x_n = \mathcal{O}(\alpha_n)$ if there exist contants C and N such that for all $n \geq N$*

$$|x_n| \leq C|\alpha_n|$$

For example, we have that

$$\frac{n+1}{n^2} = \mathcal{O}(\frac{1}{n})$$
$$Cn^3 + n - 1 = \mathcal{O}(n^3)$$
$$n(n!) + n! - 1 = \mathcal{O}(n(n!))$$

Note that it is also true that

$$Cn^3 + n - 1 = \mathcal{O}(n(n!)).$$

To fix this up, we introduce (somewhat less common) big Θ notation:

Definition 2. *Let $(x_n)_{n\in\mathbb{N}}$ and $(\alpha_n)_{n\in\mathbb{N}}$. We say that $x_n = \Theta(\alpha_n)$ if there exist contants $0 < c < C$ and N such that for all $n \geq N$*

$$c|\alpha_n| \leq |x_n| \leq C|\alpha_n|$$

We have that

$$n(n!) \leq n(n!) + n! - 1 \leq 2n(n!), \quad \forall n \in \mathbb{N}$$

so

$$n(n!) + n! - 1 = \Theta(n(n!))$$

similarly

$$Cn^3 \leq Cn^3 + n - 1 \leq (C+1)n^3, \quad \forall n \in \mathbb{N}$$

so $Cn^3 + n - 1 = \Theta(n^3)$.

1.5 Horner's algorithm

Question 8. *How many elementary operations (addition, multiplication etc.) are needed to evaluate a polynomial of the form*

$$p(z) = a_0 + a_1 z + \ldots + a_n z^n,$$

where z and $a_0, \ldots, a_n$ are given (non-zero) real numbers?

Answer 6. *The answer depends on the algorithm used. One simple way to go is computing all expressions $a_k z^k$ and then summing them together. Computing an expression of the form $a_k z^k$ requires k multiplication operations. Altogether the number of multiplication operations needed is*

$$0 + 1 + \cdots + n = \frac{n(n+1)}{2}.$$

We also need n addition operations. Altogether we need $n + \frac{n(n+1)}{2} = \Theta(n^2)$ operations.

There is a trick known as Horner's algorithm which requires only $\Theta(n)$ operations. This trick is based on the remainder theorem:

Theorem 3 (Remainder theorem)**.** *Let F be a field, $p \in F_n[z]$ and $z_0 \in F$. There exists $r \in F$ and $q \in F_{n-1}[z]$ such that*

$$p(z) = (z - z_0)q(z) + r \tag{4}$$

Remark In the theorem $F_n[z]$ is the space of polynomials of degree $\leq n$, with coeffecients in F.

Remark Note that $p(z_0) = r$. Thus, to evaluate a polynomial it is sufficient to compute the decomposition (13). The idea of Horner's algorithm is that this can be done with linear complexity.

Proof. Let $p(z) = a_n z^n + \ldots + a_1 z + a_0$, our goal is to find coeffecients $b_0, \ldots, b_{n-1}$ for q and r in F such that (13) holds: that is

$$\sum_{k=0}^{n} a_k z^k = p(z) = (z - z_0) \sum_{k=0}^{n-1} b_k z^k + r = \sum_{k=1}^{n} b_{k-1} z^k - z_0 \sum_{k=0}^{n-1} b_k z^k + r$$

by equating coefficients of each z^k this gives us the equations

$$\begin{aligned} a_n &= b_{n-1} \\ a_{n-1} &= b_{n-2} - z_0 b_{n-1} \\ a_{n-2} &= b_{n-3} - z_0 b_{n-2} \\ &\vdots \\ a_1 &= b_0 - z_0 b_1 \\ a_0 &= -z_0 b_0 + r \end{aligned}$$

Solving these equations recursively from top to bottom we see that they have a unique solution:

$$\begin{aligned} b_{n-1} &= a_n \\ b_{n-2} &= a_{n-1} + z_0 b_{n-1} \\ b_{n-3} &= a_{n-2} + z_0 b_{n-2} \\ &\vdots \\ b_0 &= a_1 + z_0 b_1 \\ r &= a_0 + z_0 b_0 \end{aligned}$$

□

It is convenient to redefine $b_{-1} := r$, so that the equations above are just

$$b_{n-1} = a_n \tag{5}$$
$$b_{k-1} = a_k + z_0 b_k, k = n-1, n-2, \ldots, 0. \tag{6}$$

These equations can be written graphically in a convenient way as shown in Figure 1 (left).

Example Evaluate $p(3)$ where

$$p(z) = z^4 - 4z^3 + 7z^2 - 5z - 2$$

The answer is 19. See Figure 1(right)

	a_n	a_{n-1}	a_{n-2}	$\ldots$	a_0
z_0		$z_0 b_{n-1}$	$z_0 b_{n-2}$	$\ldots$	$z_0 b_0$
	b_{n-1}	b_{n-2}	b_{n-3}	$\ldots$	$\boxed{b_{-1}}$

	1	−4	7	−5	−2
3		3	−3	12	21
	1	−1	4	7	19

Figure 1: Taken from our course book, page 112

Complexity of Horner's algorithm In (6) we have n equations, each involving one addition and one multiplication operation. Overall the complexity of computing $p(z_0)$ is $\Theta(n)$.

The complete Horner algorithm Horner's algorithm gives us not only the value of p at z_0, but also the coefficients of q in the decomposition in (13). This can be useful for other things. For example, say we are trying to find all roots of p, and we found a root z_0 (we will discuss solution of non-linear equations soon). Then we can compute q, and we get a polynomial of one degree lower satisfying

$$p(z) = (z - z_0) q(z).$$

We can now find a root z_1 of q, which is also a root of p. We then divide q by $z - z_1$, and continue in this fashion until we get all roots.

Another important example: say we are given the coefficients $a_0, \ldots, a_n$ of p

$$p(z) = a_0 + a_1 z + \ldots + a_n z^n.$$

For a given point z_0, there exist $c_0, \ldots, c_n$ such that

$$p(z) = c_0 + c_1(z - z_0) + \ldots + c_n(z - z_0)^n.$$

We'd like to compute $c_0, \ldots, c_n$. Note that this is equivalent to computing the derivatives at z_0 since

$$c_k = \frac{p^{(k)}(z_0)}{k!}.$$

Note also that we already know how to compute $c_0 = p(z_0)$ using Horner's algorithm. Next, note that we also know the coeffecients of q from the decomposition in (13), and q satisfies

$$c_1(z - z_0) + c_2(z - z_0)^2 + \ldots c_n(z - z_0)^n = p(z) - c_0 = p(z) - r = q(z)(z - z_0)$$

so

$$q(z) = c_1 + c_2(z - z_0) + \ldots c_n(z - z_0)^{n-1}$$

We can now use Horner's algorithm to compute

$$q(z) = (z - z_0)\bar{q}(z) + \bar{r}$$

and we will get that

$$\bar{r} = q(z_0) = c_1, \bar{q}(z) = c_2 + \ldots c_n(z - z_0)^{n-2}.$$

We continue applying Horner's algorithm recursively until we obtain all coeffecients $c_0, \ldots, c_n$.

Question 9. *Compute the coefficeints $c_0, \ldots, c_n$ such that*

$$p(z) = z^4 - 4z^3 + 7z^2 - 5z - 2$$

can be written as

$$p(z) = c_0 + c_1(z - 3) + \ldots + c_n(z - 3)^n.$$

Answer 7. *The answer is $p(z) = 19 + 37(z - 3) + 25(z - 3)^2 + 8(z - 3)^3 + (z - 3)^4$ See Figure 2.*

2 Solving non-linear equations

2.1 Bisection method

Example Find a solution to the equation $e^x = \sin x$. Our first step is converting to standard form $f(x) = e^x - \sin x = 0$. Next, we find an interval $[a, b]$ in which there must be a solutions because $sign(f(a)) = -sign(f(b))$. For example take $b = 0$ for which $f(x) = 1$ and $a = -\frac{3\pi}{2}$ for which $e^a < e^0 = 1$ and $-\sin(a) = -1$. We then use the bisection method:

Biscetion method In the bisection method we wish to find a root of a continuous function f defined on an interval $[a, b]$ where $sign(f(a)) \neq sign(f(b))$. We do this using the following recursive procedure:

Step (0) Set $a_0 = a, b_0 = b$ and $c_0 = \frac{a_0 + b_0}{2}$

Step (k) Assume a_k, b_k, c_k is defined, we define

If $f(c_k) = 0$ we found a root.

If $sign(f(a_k)) = sign(f(c_k))$ set $a_{k+1} = c_k$ and $b_{k+1} = b_k$. We have that $sign(f(a_{k+1})) = sign(f(a_k)) \neq sign(f(b_k)) = sign(f(b_{k+1}))$. and the interval $[a_{k+1}, b_{k+1}]$ became twice as small

$$b_{k+1} - a_{k+1} = b_k - \frac{a_k + b_k}{2} = \frac{b_k - a_k}{2}$$

Otherwise we know that $sign(f(b_k)) = sing(f(c_k))$ set $b_{k+1} = c_k$ and $a_{k+1} = a_k$. We have that $sign(f(b_{k+1})) = sign(f(b_k)) \neq sign(f(a_k)) = sign(f(a_{k+1}))$. and the interval $[a_{k+1}, b_{k+1}]$ again became twice as small

$$b_{k+1} - a_{k+1} = \frac{a_k + b_k}{2} - a_k = \frac{b_k - a_k}{2}$$

Set $c_{k+1} = \frac{a_{k+1}+b_{k+1}}{2}$.
Remark $b_k - a_k = 2^{-k}(b_0 - a_0)$

Theorem 4. *Let f be a continuous function defined on $[a, b]$ such that $sign(f(a)) \neq sign(f(b))$. Let a_k, b_k, c_k be the sequences produced by the bisection method and assume f is not identically zero on a_k, b_k, c_k. Then c_k converge to a limit $L \in [a, b]$ which is a root of f, and*

$$|L - c_k| \leq 2^{-(k+1)})(b - a)$$

	1	−4	7	−5	−2
3		3	−3	12	21
	1	−1	4	7	19
3		3	6	30	
	1	2	10	37	
3		3	15		
	1	5	25		
3		3			
	1	8			

Figure 2: taken from our course book

Proof. The sequence a_k is weakly monotonely increasing and bounded above by b_0, it thus has a limit L_a. The sequence b_k is weakly monotonely decreasing and has a limit L_b. We have

$$L_b - L_a = \lim_{k\to\infty} b_k - a_k = \lim_{k\to\infty} 2^{-k}(b_0 - a_0) = 0$$

so the two limits are the same. Since $signf(a_k) \neq sign(f(b_k))$ we must have $f(L) = 0$ from continuity.

For fixed k, note that $L \in [a_k, b_k]$ and so

$$|L - c_k| \leq \frac{b_k - a_k}{2} = 2^{-(k+1)}(b - a)$$

□

Lesson 5

Example If f is a continuous function on $[a,b] = [50, 63]$ and $f(a) < 0 < f(b)$, how many steps do we need to compute a root with relative accuracy of 10^{-12}:

We want to find k so that

$$\frac{|L - c_k|}{|L|} \leq 10^{-12}$$

Since $L \geq 50$ we have for fixed k that

$$\frac{|L - c_k|}{|L|} \leq \frac{|L - c_k|}{50} \leq 2^{-(k+1)}(b-a)/50 = 2^{-(k+1)}\frac{13}{50}$$

so it is sufficient of find k such that

$$2^{-(k+1)}\frac{13}{50} \leq 10^{-12}$$

or equivalently find k large enough so that

$$2^{k+1} \geq \frac{13}{50}10^{12}$$

so

$$k + 1 \geq \lceil log_2\left(\frac{13}{50}10^{12}\right)\rceil = 38$$

Psuedo-code for the bisection method

Here is some psuedo-code for the bisection method. Note that

1. We don't store all the a_k, b_k, c_k but only the current a, b, c.
2. We need stopping conditions, unlike in math, we don't want the recursive process to run forever... we also don't expect to find a root, only a number c for which $f(c) < \epsilon$ or $|c - r| \leq b - a \leq \delta$. These ϵ, δ should be chosen to be safely larger than machine precision.

Input: a, b, δ, ϵ
Output: a, b, c
$c \leftarrow (a + b)/2$
$f_a \leftarrow f(a)$
$f_b \leftarrow f(b)$
$f_c \leftarrow f(c)$
while $|f_c| > \epsilon$ and $b - a > \delta$ **do**
  if $sign(f_c) = sign(f_a)$ **then**
    $a \leftarrow c$
    $f_a \leftarrow f_c$
  else
    $b \leftarrow c$
    $f_b \leftarrow f_c$
  end if
  $c \leftarrow (a + b)/2$
  $f_c \leftarrow f(c)$
end while

2.2 Newton's Algorithm

Newton's algorithm or Newton-Raphson algorithm, is an important algorithm for computing a root of a function $f : \mathbb{R} \to \mathbb{R}$. We will assume $f \in C^2(\mathbb{R})$ and f has a root r. Assume we start with some initial guess $x_0 \in \mathbb{R}$ for the root and we want to improve it. We consider the first order Taylor expansion around x_0:

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

Since we know how to solve linear equations, we can try to improve our guess x_0 by guessing x_1 to be the solution of the equation

$$L(x_1|x_0) = 0$$

which explicitly gives us

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

note we need to assume $f'(x_0) \neq 0$. If $f'(x_0) = 0$ we define $x_1 = x_0$. We can continue defining x_{n+1} iteratively as the solution of $L(x_{n+1}|x_n) = 0$, again assuming that $f'(x_n) \neq 0$ and otherwise setting $x_{n+1} = x_n$.

Question: Does this procedure converge?

Answer: Certainly not all the time. For example, if the derivative at some point is zero, we are stuck, but this will not occur very often. There are more natural examples where this procedure will not converge to a root: one example can be found in the course book page 84, Figure 3.5. Here is another one:

Example 1 Assume that f is some function in $C^2(\mathbb{R})$ such that $f(0) = 0$, and $f(x) > 0, f'(x) < 0$ for all $x \in [1, \infty)$. Moreover assume that $\lim_{x\to\infty} f(x) = 1$. Assume $x_0 > 1$. Then

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} > x_0 > 1$$

Using this argument recursively we see that x_k will be a monotonely increasing sequence and so will not converge to a root.

There are some classes of functions for which Netwton's method does converge globally, independently of the starting point.

Example 2 Assume that $f : \mathbb{R} \to \mathbb{R}$ is twice differentiable, has a root and $f'(x) > 0, f''(x) > 0$ for all $x \in \mathbb{R}$. For example $f(x) = e^x - 6$. Then f has a unique root, and Newton iterations, initialized from any point x_0 will converge. To see this, first note that since f is monotonely increasing it cannot have more than one root. Next, note that for $n \geq 0$, using Taylor expansion around x_n we see that there exists some ξ such that

$$f(x_{n+1}) = L(x_{n+1}|x_n) + \frac{1}{2}f''(\xi)(x_{n+1} - x_n)^2 = \frac{1}{2}f''(\xi)(x_{n+1} - x_n)^2 > 0.$$

Since f is monotonely increasing it follows that $x_{n+1} > r$ for all $n \geq 0$. Next note that for all $n \geq 1$,

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

we see that x_n is monotonely decreasing and bounded by below, and hence converges to some limit L. From (7) at the limit $n \to \infty$ we see that

$$L = L - \frac{f(L)}{f'(L)}$$

so $f(L) = 0$. Uniqueness of the root implies $L = r$.

More importantly, under pretty general conditions, when the starting point is close enough to the true solution, we achieve very fast convergence. The intuition for this is that the linear approximation is accurate up to a quadratic term which is very small near the root. We will soon see these two results, but first let's start with an example showing the rapid convergence:

Example Consider the function $f(x) = e^x - 6$. The first five digits of the root is given by $r = log(6) = 1.7918$. Say we search for this root, starting from a pretty close point $x_0 = 2$. The newton iteration for f are given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{\exp(x_n) - 6}{\exp(x_n)}$$

Applying this procedure 4 times in Matlab we get (showing only five digits)

$$x_0 = 2, x_1 = 1.8120, x_2 = 1.7920, x_3 = 1.7918, x_4 = 1.7918$$

with the error $e_n = |x_n - r|$ being (showing only leading digit)

$$e_0 = 0.2, e_1 = 2 \times 10^{-2}, e_2 = 2 \times 10^{-4}, e_3 = 2 \times 10^{-8}, e_4 = 4 \times 10^{-16}$$

Our local and global convergence results both require the following lemma:

remark In the following we use the convention that whenever $a < b$, we define $(b, a) := (a, b)$. We say that r is a root of f if $f(r) = 0$. a root r of f is simple if $f'(r) \neq 0$.

Theorem 5. *Let $f \in C^2(\mathbb{R})$ be a function with a* simple *root $r \in \mathbb{R}$. Then there exists $\delta > 0$ and $C > 0$ such that for every Newton iterations starting from $x_0 \in [r-\delta, r+\delta]$, the sequence x_n obtained from the Newton iterations satisfies that $|x_n - r|$ monotonely decrease to zero, and*

$$|x_{n+1} - r| \leq C(x_n - r)^2$$

Proof. Let (x_n) be the sequence of Newton iterates starting from a point x_0. Next note that for every fixed n, there exists $\xi = \xi(n) \in (r, x_n)$ such that

$$0 = f(r) = L(r|x_n) + \frac{1}{2}f''(\xi)(x_n - r)^2.$$

Since $L(x_{n+1}|x_n) = 0$ we have

$$f'(x_n)(x_{n+1} - r) = L(x_{n+1}|x_n) - L(r|x_n) = -L(r|x_n) = \frac{1}{2}f''(\xi)(x_n - r)^2 \tag{7}$$

Since $f'(r) \neq 0$, we can choose some $\delta_0 > 0$ small enough so that $f'(x) \neq 0$ on $[r-\delta_0, r+\delta_0]$. Set

$$m = \min_{|x-r|\leq\delta_0} |f'(x)|, \text{ and } M = \max_{|x-r|\leq\delta_0} |f''(x)|$$

then using (7) we obtain that whenever some x_n is in $[r-\delta_0, r+\delta_0]$, then since every ξ between r and x_n will be in $[r-\delta_0, r+\delta_0]$ as well, we obtain

$$|x_{n+1} - r| = \frac{1}{2}\left|\frac{f''(\xi)}{f'(x_n)}\right|(x_n - r)^2 \leq \frac{M}{2m}(x_n - r)^2 \tag{8}$$

To ensure monotonicity we choose δ to be smaller than δ_0 and $\frac{1}{2}(\frac{M}{2m})^{-1}$. Then for all x_n with $|x_n - r| < \delta$ we obtain that

$$|x_{n+1} - r| < \frac{M}{2m}\delta|x_n - r| < \frac{1}{2}|x_n - r| \tag{9}$$

If $|x_0 - r| < \delta$, we have that $|x_1 - r| < |x_0 - r| < \delta$ and so we can apply the formula (9) to achieve that $|x_n - r|$ is monotonely decreasing to zero. In particular each x_n is less that δ_0 away from r and so satisfies (8) . □

Lesson 6

Convergence rate The convergence rate of Newton's method near a minimum is quadratic, meaning

$$|x_{n+1} - r| \leq C(x_n - r)^2$$

in comparison, the convergence rate of the bisection method was linear, in the sense that

$$|b_{n+1} - a_{n+1}| \leq \frac{1}{2}||b_n - a_n|$$

and the approximate solution c_n at the n-th step satisfies $|c_n - r| \leq |b_n - a_n|$.

Definition 3. *Let x_n be a sequence converging to L. We say that x_n has convergence rate $\alpha > 0$ if there exists $C, N > 0$ such that for all $n > N$,*

$$|x_{n+1} - L| \leq C|x_n - L|^\alpha$$

2.3 Newton-like method without derivatives

Newton interations require evaluations of the function f and its derivatives. Sometimes we can't, or don't want to, evaluate the derivatives. There are several methods which imitate Newton and can achieve similar convergence results without derivative evaluation. We will consider the secant method.

In the Secant method we use two starting points x_0, x_1. The iteration we use to defined x_{n+1} based on x_n, x_{n-1} are based on approximating the derivative at x_n by

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

this gives us the update rule

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

This method can be shown to converge if x_0, x_1 are chosen 'close enough' to a simple root r of f. Interestingly, the order of convergence of this method is the golden ration

$$\alpha = \frac{1 + \sqrt{5}}{2} \approx 1.62$$

which is not as good as Newton. However here we need a single new function evaluation at each iteration. With two evaluations the Secant method can beat Newton since $2\alpha > 2$:

$$|x_{n+2} - r| \leq C|x_{n+1} - r|^\alpha \leq C^2|x_n - r|^{2\alpha}$$

Generalization to higher dimensions Assume $f : \mathbb{R}^d \to \mathbb{R}^d$ is continuously twice differentiable and has a root $r \in \mathbb{R}^d$. Given a point $x_n \in \mathbb{R}^d$, we can define x_{n+1} as the solution of the linear approximation of f around x_n, that is

$$L(x|x_n) = f(x_n) + D_f(x_n)(x - x_n)$$

The solution x_{n+1} for the equation $L(x|x_n) = 0$ is given by

$$-f(x_n) = D_f(x_n)(x_{n+1} - x_n) \tag{10}$$

so

$$x_{n+1} = x_n - D_f^{-1}(x_n)f(x_n) \tag{11}$$

Note that in analogy to the scalar case where we needed $f'(x_n) \neq 0$, here we need $D_f(x_n)$ to be non-singular. We can obtain quadratic convergence for the multivariate Newton's method in a small neigborhood of a root r, providing that $D_f(r)$ is non-singular.

Remark: It is typically faster and more stable not to solve a linear equation $Ax = b$ by inverting A and setting $x = A^{-1}b$, but rather by solving directly using Gauss elimination. Therefore it would be better in practices to compute $x_{n+1} - x_n$ from the equation (10), rather than computing x_{n+1} from (11).

Newton vs. bisection Bisection advantages: Global convergence, only needs continuity.

Newton advantages: Quadratic convergence near root, generalizable to higher dimensions.

2.4 Contractive mapping theorem

The update rule in Newton's method for finding a root of a differentiable function f is given by

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

Define a function F as follows

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

The function F is defined for points where the derivative of f doesn't vanish. Then Newton iterations are given by $x_{n+1} = F(x_n)$. If r is a point where $f'(r) \neq 0$, then

$$F(r) = r \text{ if and only if } f(r) = 0.$$

A point satisfying $F(r) = r$ is called a *fixed point*. Algorithms which compute sequences $x_{n+1} = F(x_n)$, and aim to find a fixed point of F at the limit are called fixed point iterations. A general helpful tool for their analysis is the contractive mapping theorem:

Definition 4. *Let (X, d_X) and (Y, d_Y) be metric spaces. A function $F : X \to Y$ is contractive, if there exists $\lambda < 1$ such that*

$$d_Y(F(x), F(y)) \leq \lambda d_X(x, y)$$

Definition 5. *A metric space (X, d) is* complete *if every Cauchy sequence $(x_n) \subseteq X$ has a limit in X.*

Remark (1) A contractive mapping is a Lipschitz mapping with Lipschitz constant smaller than one.

(2) If you feel uncomfortable with complete metric spaces, you can just think of closed sets $X \subseteq \mathbb{R}^d$, with the metric

$$d(x, y) = \|x - y\|$$

A Cauchy sequence in $\mathbb{R}^d$ has a limit. If $X \subseteq \mathbb{R}^d$ is closed then every Cauchy sequence in X has a limit in X.

(3) If F is continuously differentiable in $[a, b]$ and $|F'(t)| \leq \lambda < 1$ for all $t \in [a, b]$, then F is a contraction since by Lagrange's theorem for every $a \leq x < y \leq b$ there exists $c \in (x, y)$ such that

$$F(x) - F(y) = F'(c)(x - y)$$

so

$$|F(x) - F(y)| = |F'(c)||x - y| \leq \lambda |x - y|$$

Theorem 6. *Let (X, d) be a complete metric space, and $F : X \to X$ a contraction with contraction factor $\lambda < 1$. Then F has a unique fixed point $r \in X$. Moreover, every sequence $(x_n)_{n=0}^{\infty} \subseteq X$ of the form*

$$x_{n+1} = F(x_n)$$

converges to r (at least) linearly

$$|x_{n+1} - r| \leq \lambda |x_n - r|$$

Proof. Let (x_n) be a sequence in X with $x_{n+1} = F(x_n)$. We show it is Cauchy and hence has a limit. For every $\epsilon > 0$ we need to show that there exists N such that for all $N < m < n$ we have $d(x_n, x_m) < \epsilon$. We will see that this is true for N large enough such that

$$\frac{\lambda^N d(x_1, x_0)}{1 - \lambda} < \epsilon.$$

We have

$$d(x_{n+1}, x_n) = d(F(x_n), F(x_{n-1})) \leq \lambda d(x_n, x_{n-1}).$$

Applying this recursively we get

$$d(x_{n+1}, x_n) \leq \lambda d(x_n, x_{n-1}) \leq \lambda^2 d(x_{n-1}, x_{n-2}) \leq \ldots \leq \lambda^n d(x_1, x_0)$$

For any $n > m \geq N$ we have

$$\begin{aligned} d(x_n, x_m) &\leq d(x_m, x_{m+1}) + d(x_{m+1}, x_{m+2}) + \ldots + d(x_{n-1}, x_n) \\ &\leq \lambda^m d(x_1, x_0) + \lambda^{m+1} d(x_1, x_0) + \ldots + \lambda^{n-1} d(x_1, x_0) \\ &= \lambda^m d(x_1, x_0)(1 + \lambda + \lambda^2 + \ldots + \lambda^{n-1-m}) \\ &\leq \lambda^N d(x_1, x_0) \sum_{k=0}^{\infty} \lambda^k \\ &= \frac{\lambda^N d(x_1, x_0)}{1 - \lambda} \end{aligned}$$

Thus we proved x_n is Cauchy and has a limit r. A Lipschitz function is in particular continuous and so

$$F(r) = \lim_{n \to \infty} F(x_n) = \lim_{n \to \infty} x_{n+1} = r$$

so r is a fixed point of F, and in particular a fixed point exists.

We next prove uniqueness of the fixed point: if r, s are two fixed points of F the

$$d(r, s) \leq \lambda d(F(r), F(s)) = \lambda d(r, s)$$

so $d(r, s) = 0$.

Finally we prove linear convergence rate

$$d(x_{n+1}, r) = d(F(x_n), F(r)) \leq \lambda d(x_n, r).$$

□

Examples

1. The function $F(x) = \frac{1}{2}x + 6$ maps the complete metric space $\mathbb{R}$ to itself and is a contraction with $\lambda = 1/2$.

   $$|F(x) - F(y)| = \frac{1}{2}|x - y|$$

   By the contractive mapping theorem it has a unique fixed point. Indeed we can solve the equation

   $$F(x) = x$$

   and obtain that this fixed point is $r = 12$. Moreover, by the contractive mapping theorem, for every $x_0 \in \mathbb{R}$, the sequence defined recursively by $x_{n+1} = F(x_n)$ will converge to r. Indeed starting from $x_0 = 0$ for example, we get

   $$x_0 = 0, x_1 = 6, x_2 = 9, x_3 = 10\frac{1}{2}, x_4 = 11\frac{1}{4}.$$

2. The restriction of the function $F(x) = \frac{1}{2}x + 6$ to $X = [0, 1]$ does not have a fixed point. The conditions of the contractive mapping theorem do not apply here because $F(X)$ is not contained in X.

3. the restriction of the same function to the interval $X = (0, 12)$ does not have a fixed point in X. The conditions of the contractive mapping theorem do not apply here because X is not a complete metric space.

4. The function $F(x) = -x$ maps $X = [-1, 1]$ to itself. It if not a contraction-it has a Lipschitz constant of 1. It has a fixed point $r = 0$, but for all $x_0 \neq r$ the fixed point iterations starting from x_0 do not converge.

5. Consider the function $F(x) = x + \frac{1}{x}$ defined on $X = [1, \infty)$. The set X is closed, and $F(X) \subseteq X$. Moreover F is 'almost a contraction'. For all $x, y \in X$ with $x \neq y$, we have that

   $$F(x) - F(y) = (x - y) + (\frac{1}{x} - \frac{1}{y}) = (x - y) + \frac{y - x}{xy} = (1 - \frac{1}{xy})(x - y)$$

   Thus

   $$|F(x) - F(y)| \leq |1 - \frac{1}{xy}||x - y|, \forall x, y \in X \tag{12}$$

   Note that $1 > 1 - \frac{1}{xy} > 0$ for all $x, y \in X$. However, we cannot replace the expression $|1 - \frac{1}{xy}|$ in (12) with any constant $\lambda < 1$. Thus the theorem does not apply. Indeed, F does not have a fixed point, as there is not solution to the equation

   $$x + \frac{1}{x} = x.$$

Lesson 7

Analysis of Newton's method via fixed point iterations Recall the following theorem which we already proved in class:

Theorem. *Let $f \in C^2(\mathbb{R})$ be a function with a* simple *root $r \in \mathbb{R}$. Then there exists $\delta > 0$ and $C > 0$ such that for every Newton iterations starting from $x_0 \in [r-\delta, r+\delta]$, the sequence x_n obtained from the Newton iterations satisfies that $|x_n - r|$ monotonely decrease to zero, and*

$$|x_{n+1} - r| \leq C(x_n - r)^2$$

Second proof under the additional assumption that f has three continuous derivatives. We will now see a simple proof using the contraction mapping theorem. This proof however will need to assume that f is *three times* continuously differentiable: let r be a simple root of f. Then r is a fixed point of F, which is defined as

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

The derivative of F at a point x is given by

$$F'(x) = 1 - \frac{[f'(x)]^2 - f(x)f''(x)}{[f'(x)]^2} = \frac{f(x)f''(x)}{[f'(x)]^2}$$

In particular $F'(r) = 0$ since $f(r) = 0$ and $f'(r) \neq 0$. Thus since f is three times continuously differentiable, we can choose a small enough δ such that for all $x \in [r-\delta, r+\delta] = X_\delta$, we have $f'(x) \neq 0$ and $|F'(x)| < \lambda$ for some $\lambda < 1$. Note that $F(X_\delta) \subseteq X_\delta$ since

$$|F(x) - r| = |F(x) - F(r)| \leq \lambda |x - r| < \delta$$

Thus by the contractive mapping theorem Newton iteration converge to the unique fixed point r.

Regarding the rate of convergence: assume $(x_n) \subseteq X_\delta$, and

$$x_{n+1} - r = F(x_n) - F(r) = [F(r) + F'(r)(x_n - r) + \frac{1}{2}F''(c)(x_n - r)^2] - F(r)$$

where c is some point between r and x_n, and we used the fact that f is three times continuously differentiable and thus F is twice continuously differentiable. Since $F'(r) = 0$ we get quadratic convergence

$$|x_{n+1} - r| \leq \frac{1}{2} \max_{q \in X_\delta} |F''(q)| |x_n - r|^2$$

□

Example: Gradient descent Here is another example of a fixed point iteration method and its analysis: Assume we want to minimize a twice continuously differntiable function $f : \mathbb{R} \to \mathbb{R}$ and let r be a strict minimum ($f'(r) = 0$ and $f''(r) > 0$). Gradient descent is a method to search for the minimum based on the update rule $x_{n+1} = F(x_n)$ where

$$F(x) = x - \alpha f'(x)$$

that is, we take a step towards the descent step of the function. $\alpha > 0$ is the *step-size*. We can prove

Theorem 7. *Let r be a strict minimum of $f \in C^2(\mathbb{R})$. Then for all small enough $\alpha > 0$ there exists $\delta > 0$ such that the gradient descent iterations converge linearly to the minimum r whenever they are initialized from x_0 which is less that δ away from r.*

Proof. The point r is a fixed point of F. Note that

$$F'(x) = 1 - \alpha f''(x)$$

for every positive α we have that

$$F'(r) < 1$$

if α is small enough so that $\alpha F''(r) < 2$ then we also have

$$F'(r) > -1$$

so $|F'(r)| < 1$. It follows that we can find some δ such that if $x \in X_\delta$ then $|F'(x)| < \lambda$ for some $\lambda < 1$. As we seen previously this implies that $FX_\delta \subseteq X_\delta$ and thus we see that iterations initialized from some $x_0 \in X_\delta$ converge linearly to a fixed point by the contractive mapping theorem. □

2.5 Polynomials and their roots

Our next topic is finding roots of polynomials.

When we discuss polynomials with real coefficients, and we are looking for real roots, we can use any one of the algorithms we saw up to now. If we are looking for complex roots, we can use the *complex Newton method* which is essentially the same as the standard Newton method, but uses the notion of a complex derivative of a function $f : \mathbb{C} \to \mathbb{C}$. A polynomial over $\mathbb{C}$ has a complex derivative which is exactly what we'd expect, that is if

$$p(z) = a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0$$

then

$$p'(z) = n a_n z^{n-1} + (n-1) a_{n-1} z^{n-2} + \ldots + a_1$$

and we can define Newton iterations as usual

$$z_{n+1} = z_n - p(z_n)/p'(z_n)$$

and again get local quadratic convergence near simple roots etc.

Reminders: roots of polynomials Let F be a field. We say a function $p : F \to F$ is a polynomial of degree n if

$$p(z) = a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0$$

where $a_n \neq 0$ and $a_j \in F, \forall j = 0, \ldots, n$.

We denote the set of polynomials with coeffecients in F of degree n by $F_n[z]$.

Theorem 8 (The fundamental theorem of algebra)**.** *Every non-constant polynomial with (real or) complex coefficients has a complex root.*

we previously discussed the remainder theorem.

Theorem 9 (Remainder theorem)**.** *Let $p \in F_n[z]$ and $z_0 \in F$. There exists $r \in F$ and $q \in F_{n-1}[z]$such that*

$$p(z) = (z - z_0) q(z) + r \tag{13}$$

Corollary 1. *z_0 is a root of a polynomial p of degree n, if and only if $p(z) = (z - z_0)q(z)$ for some polynomial q of degree $n - 1$.*

Proposition 3. *If $p \in \mathbb{R}_n[x], n \geq 2$ then there exists $u, v \in \mathbb{R}$ and $q \in \mathbb{R}_{n-2}[x]$ such that*

$$p(x) = (x^2 + ux + v) q(x).$$

Proof. If p has two real roots x_0, x_1 then by applying the corollary twice we see that there exists $q \in \mathbb{R}_{n-2}[x]$ such that

$$p(x) = (x - x_0)(x - x_1) q(x) = (x^2 - (x_0 + x_1)x + x_0 x_1) q(x).$$

Otherwise, p has at least one non-real root z_0. We note that if z_0 is a root then so is $\bar{z}_0$ because if

$$0 = p(z_0) = a_0 + a_1 z_0 + \ldots a_n z_0^n$$

then

$$0 = \overline{p(z_0)} = a_0 + a_1 \bar{z_0} + \ldots a_n \bar{z_0}^n = p(\bar{z}_0).$$

It follows that there exists $q \in \mathbb{C}_{n-2}[z]$ such that

$$p(z) = (z - z_0)(z - \bar{z}_0) q(z) = (z^2 - (z_0 + \bar{z_0})z + |z_0|^2) q(z).$$

as required. Note that q multiplied by a real polynomial yields another real polynomial and thus q is real too.

□

Lesson 8

Bairstow's method Bairstow's method is a method for computing two roots of a real polynomial simultaneously. It is based on three facts:

1. A real polynomial p of degree $n \geq 2$ can always be divided by a real quadratic polynomial of the form $x^2 - u_* x - v_*$ as shown in Proposition 3
2. If we can find a quadratic polynomial $x^2 - u_* x - v_*$ dividing p then the two (real or complex) roots of the quadratic polynomials are also roots of p. We have a formula to compute these roots which we know from high-school (only this time, this formula is well defined even which $b^2 - 4ac < 0$ since we allow complex numbers).
3. For *any* u, v, we can divide p by $x^2 - ux - v$ with remainder: that is, there exist unique $b_j = b_j(u, v) \in \mathbb{R}, j = 0, \ldots, n$, such that the degree $n-2$ polynomial
$$q(x) = b_n x^{n-2} + \ldots + b_3 x + b_2$$
satisfies
$$p(x) = (x^2 - ux - v)(b_n x^{n-2} + \ldots + b_3 x + b_2) + b_1(x - u) + b_0. \tag{14}$$
Since this decomposition is defined uniquely, for fixed u, v, we can think of $b_j, j = 0, \ldots, n$ as functions of u, v. Our goal is to find (u_*, v_*) such that
$$b_1(u_*, v_*) = 0$$
$$b_0(u_*, v_*) = 0.$$
We will do this using Newton's method for functions from $\mathbb{R}^2$ to $\mathbb{R}^2$.

Let us now write out our equations explicitly. Recall that for $f : \mathbb{R}^d \to \mathbb{R}^d$ Newton iterations are given by
$$x_{k+1} = x_k - D_f^{-1}(x_k)(f(x_k))$$
Thus we need to know how to evaluation $f(u, v) = (b_0(u, v), b_1(u, v))$ and its differential.

Returning to the equation (14) we get (defining $b_{n+1} = 0 = b_{n+2}$)
$$\begin{aligned}
\sum_{k=0}^{n} a_k x^k = p(x) &= \sum_{k=2}^{n} b_k x^k - u \sum_{k=2}^{n} b_k x^{k-1} - v \sum_{k=2}^{n} b_k x^{k-2} + b_1(x - u) + b_0 \\
&= \sum_{k=2}^{n} b_k x^k - u \sum_{k=1}^{n-1} b_{k+1} x^k - v \sum_{k=0}^{n-2} b_{k+2} x^k + b_1 x - b_1 u + b_0 \\
&= \sum_{k=0}^{n} b_k x^k - u \sum_{k=0}^{n-1} b_{k+1} x^k - v \sum_{k=0}^{n-2} b_{k+2} x^k \\
&= \sum_{k=0}^{n} \left(b_k - u b_{k+1} - v b_{k+2}\right) x^k
\end{aligned}$$

So by equating coeffecients we get a recursive formula of obtaining b_k from b_{k+1}, b_{k+2} via $b_{n+2} = 0 = b_{n+1}$ and
$$b_k = a_k + u b_{k+1} + v b_{k+2}, k = n, n-1, \ldots, 1, 0 \tag{15}$$

By applying this formula recursively we obtain $b_1(u, v)$ and $b_0(u, v)$ so we see how to evaluate our function. To perform Newton iterations we also need to evaluate the differential of the function. Denote
$$c_k = \frac{\partial b_k}{\partial u}, d_k = \frac{\partial b_k}{\partial v} \text{ for } 0 \leq k \leq n$$

Taking the derivative of (15) according to u we get that $c_{n+1}, c_{n+2} = 0$ and
$$c_k = b_{k+1} + u c_{k+1} + v c_{k+2}, 0 \leq k \leq n$$

Taking the derivative of (15) according to v we get that
$$d_k = b_{k+2} + u d_{k+1} + v d_{k+2}$$

we can then prove

Lemma 2. *For all $k = 0, 1, \ldots, n+1$ we have $d_k = c_{k+1}$.*

Proof. We know that $d_{n+1} = 0$ since $b_{n+1} = 0$ and so $d_{n+1} = c_{n+2}$. Note that $d_n = 0 = c_{n+1}$ as well. We proved the claim for $k = n+1, n$. Now we can prove by induction that if $0 \leq k \leq n-1$ and $d_{k+1} = c_{k+2}, d_{k+2} = c_{k+3}$ then

$$d_k = b_{k+2} + ud_{k+1} + vd_{k+2} = b_{k+2} + uc_{k+2} + vc_{k+3} = c_{k+1}$$

□

We can now express the differential of our function f at a point u, v as

$$d_f(u,v) = \begin{pmatrix} \frac{\partial b_0}{\partial u}(u,v) & \frac{\partial b_0}{\partial v}(u,v) \\ \frac{\partial b_1}{\partial u}(u,v) & \frac{\partial b_1}{\partial v}(u,v) \end{pmatrix} = \begin{pmatrix} c_0(u,v) & c_1(u,v) \\ c_1(u,v) & c_2(u,v) \end{pmatrix}$$

Thus we have seen how to evaluate f and its differential, so we know how to perform Newton iterations.

Solving non-linear equations is NP-hard How difficult is it so solve non-linear equations? The bisection methods is globally convergent. We haven't seen any root finding methods with global convergence for functions $f : \mathbb{R}^n \to \mathbb{R}^n, n > 1$. Are there such methods? We claim that finding global convergence methods with polynomial complexity in n is not likely, since this will in particular solve NP hard problems. If you are not familiar with this concept, you can think of these as a class of problems for which there is no known polynomial time algorithm, and there is good reason to assume that we will never find one.

The subset-sum problem the subset-sum problem is an NP-hard problem: here we are given n integers $k_1, \ldots, k_n$ (repetitions are allowed), each represented with B bits, and an integer T, and our task is to check whether there exists a subset $S \subseteq \{1, 2, \ldots, n\}$ such that

$$\sum_{s \in S} k_s = T$$

for example, if $S = -4, -3, 2, 2, 5$ and $T = 0$ then there exists a subset $S = \{-4, 2, 2\}$ which sums to $T = 0$. For $T = 10$ there is no such subset.

One way to solve this problem is by going over all possible subsets and checking if they provide a solution: however the complexity of this grows exponentially in n, B. In general, there is no known algorithm for solving this problem with polynomial runtime, i.e, with less that $\leq (nB)^k$ operations, for some fixed k. Indeed this problem is known to be NP-hard...

Let us now show how to model the subset sum problem as a non-linear equation: given $L, k_1, \ldots, k_n \in \mathbb{Z}$ and subset $S \subseteq \{1, \ldots, n\}$. Let x^S be an n-dimensional vector with $x_i^S = 1$ if $i \in S$ and $x_i^S = 0$ otherwise. Then

$$\sum_{s \in S} k_s = \sum_{j=1}^{n} x_S(j) k_j$$

It follows that there exists S such that

$$\sum_{s \in S} k_s = T$$

if and only if there exist $x_j \in \{0, 1\}, j = 1, \ldots, n$ such that $\sum_{j=1}^n x_j k_j = T$. In other words subset sum can be solved by solving the following equations

$$x_j(1 - x_j) = 0, j = 1, \ldots, n \tag{16}$$

$$\sum_{j=1}^{n} x_j k_j = L \tag{17}$$

we conclude that having an efficient general purpose algorithm with global convergence for high-dimensional non-linear equations is unlikely.

Summary: non-linear equations We discussed Newton's method and bisection method for scalar equations. Also the secant method. We discussed Newton in high dimensions. We saw how to find roots of a real polynomials using Bairstow's algorithm, and finally we showed that NP hard problems can be formulates as non-linear equations.

Lesson 9

3 Numerical Linear Algebra

3.1 Solving linear equations

In the following we discuss algorithms for solving square linear equations $Ax = b$ where $A \in \mathbb{R}^{n\times n}$ is non-singular. We will review Gauss elimination and the closely related topic of LU factorization.

Easy linear equations When A is diagonal $A = D$, the equation $Dx = b$ can be solved in $\mathcal{O}(n)$ elementary operations.

When $A = U$ is *upper-triangular*, meaning that $A_{ij} = 0$ if $i > j$, then we can solve the equation $Ux = b$ in $\mathcal{O}(n^2)$ operations via

$$\begin{aligned} U_{nn}x_n &= b_n \\ U_{n-1,n-1}x_{n-1} + U_{n-1,n}x_n &= b_{n-1} \\ U_{n-2,n-2}x_{n-2} + U_{n-2,n-1}x_{n-1} + U_{n-2,n}x_n &= b_{n-2} \\ &\vdots \end{aligned}$$

we can see that the first equation gives us $x_n = b_n/U_{n,n}$ second equation then gives us x_{n-1} since we known x_n, and we can continue recursively. All in all we need $\mathcal{O}(n^2)$ operations.

Similarly, if $A = L$ is lower-triangular we can solve $Lx = b$ in $\mathcal{O}(n^2)$ operations.

If $A = LU$ where L is lower-triangular and U is upper-triangular, we can solve $LUx = b$ via solving

$$Ux = z \tag{18}$$
$$Lz = b \tag{19}$$

for $x, z \in \mathbb{R}^n$. Note that if A is non-singular so are L, U, and we can solve this equation by first solving for z and then solving for x, the complexity is still $\mathcal{O}(n^2)$.

The purpose of this discussion is showing that Gauss-elimination gives us a decomposition of A of the form

$$PA = LU$$

where P is a permutation matrix, L is lower-triangular and U is upper-triangular. In the following we will give a short background on permutation matrices, then show how LU decompositions are obtained using Gauss decomposition. Afterwards we will discuss why this way of viewing Gauss elimination can be helpful.

Permutation matrices A permutation is a bijective mapping $\sigma : \{1, \dots, n\} \to \{1, \dots, n\}$. A permutation σ enduces a linear map $\hat{\sigma} : \mathbb{R}^n \to \mathbb{R}^n$ which permutes the coordinates according to σ

$$[\hat{\sigma}(v_1, \dots, v_n) = (v_{\sigma(1),\dots,v_\sigma(n)}), \quad i = 1, \dots, n.$$

The matrix representing the linear operator $\hat{\sigma}$ is the matrix P^σ defined by $P^\sigma_{\sigma(j),j} = 1$ for all $j = 1, \dots, n$ and all other coordinates P^σ_{ij} are zero. Such a matrix is called a permutation matrix. To see this note that

$$\hat{\sigma}(e_j) = e_{\sigma(j)} = P^\sigma(e_j)$$

When P_σ multiplies a $n \times n$ matrix A from the left, the i-th row of A becomes the $\sigma(i)$ row of $P_\sigma A$. Thus, when we wise to swap the i and j rows, this correspondes to the permutation $\sigma(i) = j, \sigma(j) = i, \sigma(k) = k, \forall k \neq i, j$ and the permutation matrix P_σ whose non-zero entries are given by

$$[P_\sigma]_{ij} = 1, [P_\sigma]_{ji} = 1, [P_\sigma]_{kk=1}, \forall k \neq i, j$$

The permutation matrix flipping the second and third row looks like this

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

A permutation matrix is orthgonal (note all its columns are orthogonal), that is $PP^T = I_n$.

Finally, a composition of permutations is a permutation, and this implies that products of permutation matrices are permutation matrices, moreover

$$P_\sigma \circ P_\tau = P_{\sigma\circ\tau}$$

and in particular $P_{\sigma^{-1}} = P_\sigma^{-1} = P_\sigma^T$.

sign and determinant of permutations Every permutation can be realized as a finite number of composition of transpositions. A transposition is a permutation τ satisfying for some fixed $i \neq j$ that $\tau(i) = j, \tau(j) = i$, and $\tau(k) = k$ for all $k \neq i, j$. If σ can be written as a composition of k transpositions then the sign of σ is defined to be $(-1)^k$. We have

$$\det(P_\sigma) = \text{sign}(\sigma)$$

Gauss elimination for obtaining U from A

Example 1. *In class we will run Gauss elimination to transform the matrix*

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 5 & 3 \\ 3 & 0 & 1 \end{pmatrix}$$

into an upper triangular matrix. We can proceed as follows

$$A^{(0)} = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 5 & 3 \\ 3 & 0 & 1 \end{pmatrix} \mapsto A^{(1)} = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 3 \\ 0 & -6 & 1 \end{pmatrix} \mapsto A^{(2)} = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 19 \end{pmatrix}$$

Note that we did not need to permute rows in this example so this makes life a bit easier. Adding the j-th row a_i times to the i-th row is done by multiplication by a matrix of the form

$$L^{(i,j)} = I_n + a_i E^{ij},$$

where E^{ij} is the matrix with $E^{ij}_{ij} = 1$ and $E^{ij}_{k\ell} = 0$ whenever $(i, j) \neq (k, \ell)$. For example, adding the second row of a matrix $B \in \mathbb{R}^{3\times 3}$ a_3 times to the third row is equivalent to multiplication by the matrix $L^{(3,2)}$ defined as

$$L^{(3,2)}B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & a_3 & 1 \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} + a_3 b_{21} & b_{32} + a_3 b_{22} & b_{33} + a_3 b_{23} \end{pmatrix}$$

Note that $L^{(i,j)}$ is lower triangular. Thus in our case we have that U is obtained by multiplying A by the three lower triangular matrices.

$$U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 6 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix} A^{(0)}$$

Next, note that

$$\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 - 2 \cdot 1 & 1 - 2 \cdot 0 & 0 - 2 \cdot 0 \\ -30 & 0- & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix}$$

so we now can obtain U by multiplying A with only two lower triangular matrices.

$$U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 6 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix} A$$

Next, note that

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 6 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -6 & 1 \end{pmatrix}$$

since applying these matrices one after another to some matrix with rows R_1, R_2, R_3 first adds the second row 6 times to the third row, and then substracts it again. by the same logic

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

and therefore

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -6 & 1 \end{pmatrix} U.$$

Finally, note that

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -6 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0+2\cdot 1 & 1+2\cdot 0 & 0+2\cdot 0 \\ 0+3\cdot 1 & -6+3\cdot 0 & 1+3\cdot 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -6 & 1 \end{pmatrix}$$

so we obtained $A = LU$, explicitly

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -6 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 19 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 5 & 3 \\ 3 & 0 & 1 \end{pmatrix}$$

Definition 6. *For $L \in \mathbb{R}^{n\times n}$ and $j \in \{1, \dots, n\}$, we say that L is a type-j lower triangular matrix if it is of the form*

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & L_{j+1,j} & 1 & 0 & 0 \\ 0 & 0 & \vdots & 0 & \ddots & 0 \\ 0 & 0 & L_{n,j} & 0 & 0 & 1 \end{pmatrix}$$

Type j matrices differ only be their entries at the indices $(i, j), i > j$. For $\vec{a} = (a_{j+1}, \dots, a_n)$ we denote

$$L^{(j)}(\vec{a}) = L^{(j)}(a_{j+1}, \dots, a_n) = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & a_{j+1} & 1 & 0 & 0 \\ 0 & 0 & \vdots & 0 & \ddots & 0 \\ 0 & 0 & a_n & 0 & 0 & 1 \end{pmatrix}$$

We now have the following

Theorem 10. *For every $n \in \mathbb{N}$, $j \in \{1, \dots, n\}$,*

1. *for every type-j matrix $L^{(j)}(a_{j+1}, \dots, a_n)$, we have that*

$$L^{(j)}(a_{j+1}, \dots, a_n) \begin{pmatrix} R_1 \\ \vdots \\ R_j \\ R_{j+1} \\ \vdots \\ R_n \end{pmatrix} = \begin{pmatrix} R_1 \\ \vdots \\ R_j \\ R_{j+1} + a_{j+1}R_j \\ \vdots \\ R_n + a_n R_j \end{pmatrix}$$

$B \in \mathbb{R}^{n\times n}$ with rows $R_1, \dots, R_n$.

2. *For every* $\vec{a}, \vec{b} \in \mathbb{R}^{n-j-1}$ *we have that*

$$L^{(j)}(\vec{a}) \cdot L^{(j)}(\vec{b}) = L^{(j)}(\vec{a} + \vec{b}),$$

and in particular $L^{(j)}(\vec{a}) \cdot L^{(j)}(-\vec{a}) = I_n$.

3. *If* $L^{(j)}, j = 1, \ldots, n$ *are each type* j *matrices, then*

$$L^{(1)} \cdot L^{(2)} \cdot \ldots \cdot L^{(n-1)} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ L_{21}^{(1)} & 1 & 0 & 0 & 0 \\ L_{31}^{(1)} & L_{32}^{(2)} & 1 & \ldots & 0 \\ \vdots & & & \ddots & \\ L_{n1}^{(1)} & L_{n2}^{(2)} & \ldots & L_{n,n-1}^{(n-1)} & 1 \end{pmatrix}$$

Let us do one more example

Question 10. *Compute an* LU *decomposition of the matrix*

$$A = \begin{pmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{pmatrix}$$

Answer 8. *Multiplying* A *from the left by* $L^{(1)} = L^{(1)}(a_{21} = -2, a_{31} = -1/2, a_{41} = 1)$ *we obtain*

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{pmatrix} = L^{(1)}A$$

Multiplying from the left again by

$$L^{(2)} = L^{(2)}(a_{32} = -3, a_{42} = 1/2)$$

we obtain

$$\begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{pmatrix} = L^{(2)}L^{(1)}A$$

Finally by multiplying by $L^{(3)} = L^{(3)}(a_{43} = 2)$ *we obtain*

$$U := \begin{pmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{pmatrix} = L^{(3)}L^{(2)}L^{(1)}A$$

We obtained the equation

$$U = L^{(3)}(a_{43})L^{(2)}(a_{32}, a_{42})L^{(1)}(a_{21}, a_{31}, a_{41})A$$

which holds if and only if

$$L^{(1)}(-a_{21}, -a_{31}, -a_{41})L^{21)}(-a_{32}, -a_{42})L^{(3)}(-a_{43})U = A$$

which holds if and only if

$$LU = A, \text{ where } L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -a_{21} & 1 & 0 & 0 \\ -a_{31} & -a_{32} & 1 & 0 \\ -a_{41} & -a_{42} & -a_{43} & 1 \end{pmatrix}$$

From this example we can deduce the following (we will not prove this formally)

Proposition 4. *Let* $A \in \mathbb{R}^{n \times n}$ *be an invertible matrix, and assume that* U *is obtained from* A *using Gauss elimination. Assume further that no row-swapping was necessary. In the* j*-th step of Gauss elimination the* j*-th row of* $\tilde{A}^{(j-1)}$ *was multiplied by some* a_{ij} *and added to the* i*-th row. Then*

$$A = LU, \text{ where } L \text{ is lower triangular and } L_{jj} = 1, L_{ij} = -a_{ij}, \forall 1 \leq j < i \leq n$$

Lesson 10

Up to now we have carefully chosen examples where Gauss elimination did not require permuting rows. Now let us see how to deal with the more general case through the following example

Question 11. *Find L, U, P (which are lower-triangular, upper triangular, and permutation matrices respectively), such that $LU = PA$, where*

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

Answer 9. *Using Gauss elimination we have*

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 2 & 1 & 1 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 2 & 1 & 1 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = U$$

In other words

$$U = P^{(2,3)} L^{(1)} P^{(1,2)} A, \text{ where } L^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix}$$

and $P^{(a,b)}$ denotes the permutation matrix which switches the a and b rows while keeping the remaining rows fixed. To get to the representation we want we need to disentangle the L from the Ps. This is done by observing that for every 3×3 matrix with rows R_1, R_2, R_3,

$$P^{(2,3)} L^{(1)} \begin{pmatrix} R_1 \\ R_2 \\ R_3 \end{pmatrix} = P^{(2,3)} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \\ R_3 \end{pmatrix} = \begin{pmatrix} R_1 \\ R_3 - 2R_1 \\ R_2 \end{pmatrix} = \tilde{L}^{(1)} P^{(2,3)} \begin{pmatrix} R_1 \\ R_2 \\ R_3 \end{pmatrix}$$

where

$$\tilde{L}^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Therefore we have that

$$U = \tilde{L}^{(1)} P^{(2,3)} P^{(1,2)} A$$

so taking the inverse of $\tilde{L}^{(1)} = L^{(1)}(-2, 0)$, and noting that $P^{(2,3)} P^{(1,2)} = P^{(1,3,2)}$, we obtain

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = L^{(1)}(2,0) U = P^{(1,3,2)} A \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 2 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

To turn this example into a general argument, we state the following lemma

Lemma 3. *If $L \in \mathbb{R}^{n \times n}$ is a lower semi-triangular matrix of type j, and $r > s > j$, then*

$$P^{(r,s)} L = \tilde{L} P^{(r,s)}$$

where $\tilde{L}$ is the type j matrix obtained from L by swapping L_{rj} and L_{sj}.

We will not prove this. The previous example will be proof enough for us. We can now prove

Theorem 11. *Let $A \in \mathbb{R}^{n \times n}$ be a matrix, then there exist upper triangular, lower triangular, and permutation matrices L, U, P, such that $PA = LU$. Moreover the diagonal elements L can be chosen to be one.*

Proof. Gauss elimination gives us that

$$U = L^{(n-1)} P^{(n-1)} \dots L^{(2)} P^{(2)} L^{(1)} P^{(1)} A$$

where each $L^{(j)}$ is a type j lower triangular matrix, and each $P^{(s)}$ swaps two indices s and $r > s$. If follows that by Lemma 3, there exist type j matrices $\hat{L}^{(j)}$ such that

$$U = \hat{L}^{(n-1)} \hat{L}^{(n-2)} \dots \hat{L}^{(1)} P^{(n-1)} \dots P^{(1)} A$$

The product $P^{(n-1)} \dots P^{(1)}$ gives a permutation matrix which we denote by P. Using Theorem 10, we know that the inverse of each type j lower trinagular matrix $\hat{L}^{(j)}$ is another lower triangular matrix $\bar{L}^{(j)}$, and so

$$\bar{L}^{(1)} \dots \bar{L}^{(n-1)} U = PA.$$

Again by Theorem 10 we have that this produce of $\bar{L}^{(j)}$ matrices is again a lower triangular matrix. □

Remark: In fact, it is true that the product and inverse of lower triangular matrices is lower triangular. What really is useful about Theorem 10 as opposed to this general property is that we have a simple formula for the inverse and product, and this gives us an algorithm for computing the LU decomposition which has no significant additional complexity in comparison to Guass elimination. We will see how to do this in practice in the following example.

Question 12. *Find lower triangular L, upper triangular U, and a permutation matrix P, all in $\mathbb{R}^{4\times 4}$, such that*

$$LU = PA, \text{ where } A = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 2 & 2 & 2 & 2 \\ 1 & 2 & 1 & 4 \\ 1 & 2 & 2 & 6 \end{pmatrix}$$

Answer 10. *Gauss elimination gives us*

$$\begin{aligned} U := \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 2 \end{pmatrix} &= P^{(3,4)} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} P^{(2,3)} \begin{pmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} A \\ &= P^{(3,4)} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -2 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} P^{(2,3)} A \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -2 & 0 & 0 & 1 \end{pmatrix} P^{(3,4)} P^{(2,3)} A \end{aligned}$$

By taking the inverse of the type-1 and type-2 matrices and then using our rule on multiplication from Theorem 10 we obtain

$$LU = PA, \text{ where } L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix} \text{ and } P = P^{(3,4)} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Indeed we can double check and verify that

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 0 \\ 2 & 2 & 2 & 2 \\ 1 & 2 & 1 & 4 \\ 1 & 2 & 2 & 6 \end{pmatrix}$$

Complexity of Gauss elimination, solving linear equations, matrix inversion few remarks:

1. We can use LU decomposition to compute the determinant of A

   $$\det(P)\det(A) = \det(L)\det(U).$$

   The determinant of P is easy to compute. It is just $\det(P) = (-1)^N$ where N is the number of transpositions used in the Gauss elimination process. The determinant of L and U is just the product of their diagonal elements. Note also that if all diagonal entries of L are 1 as in our construction, then $\det(L) = 1$. Thus once we have an LU decomposition of A we can compute $\det(A)$ with linear complexity in n. But how long would it take to compute the LU decomposition?

2. Step k of Gauss elimination involves finding a non-zero entry $j \geq k$ in the k-th column and swapping the j-th row with the k-th row. This can be done in $n-k+1$ steps. We then add the k-th row to the rows under it with an appropriate product which takes $2(n-k+1)(n-k)$ operation. The overall complexity of doing this for $k = 1, \ldots, n-1$ is

$$\begin{aligned}\sum_{k=1}^{n-1} 2(n-k+1)(n-k) + (n-k+1) &= \sum_{j=1}^{n-1} 2j(j+1) + j + 1 \\ &= \sum_{j=1}^{n-1} 2j^2 + 3j + 1 \\ &= 2\sum_{j=1}^{n} j^2 + 3\sum_{j=1}^{n} j - n^2 - 3n + 1 \\ &= \frac{2}{6}n(n+1)(2n+1) + O(n^2) = \frac{4}{6}n^3 + O(n^2)\end{aligned}$$

where we used the fact (can be proved by induction) that

$$\sum_{j=1}^{n} j^2 = \frac{j(j+1)(2j+1)}{6}$$

so the complexity is then $\mathcal{O}(n^3)$ and

$$\lim_{n\to\infty} \frac{\text{operations to decompose } n \times n \text{ matrix}}{n^3} = 2/3$$

Once this procedure is carried out we will have a decomposition $PA = LU$ or $A = P^T LU$, giving us an equation we can solve in $\mathcal{O}(n^2)$.

3. While the complexity of performing Gauss-elimination to A is $\mathcal{O}(n^3)$. Solving $LUx = b$ after the decomposition is carried out has complexity $\mathcal{O}(n^2)$ and as a result the overall complexity of solving $Ax = b$ is $\mathcal{O}(n^3)$. What happens if we need to solve the equation $Ax = b$ for m different values $b_i, i = 1, \ldots, m$, but A is fixed? We can factorize $PA = LU$ once using Gauss elimination, and then solve $P^T LUx = b_i$ for $i = 1, \ldots m$, so the complexity is $\mathcal{O}(n^3 + mn^2)$. If we would do Gauss elimination every time we would have complexity of $\mathcal{O}(mn^3)$.

   For example, if we want to compute the inverse of A we can do this by solving n linear equations involving the columns of $A^{-1} = (c_1|c_2|\ldots|c_n)$, that is

$$Ac_j = e_j, j = 1, \ldots, n$$

   we can factorize A onces in $\mathcal{O}(n^3)$ and then solve the n equations $P^T LUc_j = e_j$ with complexity $\mathcal{O}(n^2)$ each for a total complexity of $\mathcal{O}(n^3)$. Of course the inverse can also be computed with similar complexity by only applying elementary operations to the rows and columns of A.

3.2 Linear equations: numerical aspects

Scaled row pivoting Consider the equation

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

whose solution is $(x, y) = (1, 1)$. If we solve this equation using Gaussian elimination we would swap the first row with the second row. However, what would we do if we replace the zero with some small positive ϵ to get the equation

$$\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

In theory, we do not need to swap rows here, but we will now see that this leads to numerical problems. This is a common phenomena, slight perturbations of problems which are impossible to solve, will generally be very difficult

to solve numerically. Returning to our example, let us see what happens if we don't swap rows, first with exact computation and then with numerical calculations. In the first Gauss elimination step we would remove the first row multiplied by ϵ^{-1} and get the equation

$$\begin{pmatrix} \epsilon & 1 \\ 0 & 1-\epsilon^{-1} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 2-\epsilon^{-1} \end{pmatrix}$$

The exact solution for this equation is

$$y = (2-\epsilon^{-1})/(1-\epsilon^{-1})$$

$$x = \epsilon^{-1}(1 - \frac{2-\epsilon^{-1}}{1-\epsilon^{-1}}) = -\frac{\epsilon^{-1}}{1-\epsilon^{-1}} \approx 1$$

when $\epsilon = 2^{-N}$ for N large enough (e.g., $N \gg 23$ in single precision) then $fl(1-\epsilon^{-1}) = -\epsilon^{-1}$ and $fl(2-\epsilon^{-1}) = -\epsilon^{-1}$ so we would get the equation

$$\begin{pmatrix} \epsilon & 1 \\ 0 & -\epsilon^{-1} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ -\epsilon^{-1} \end{pmatrix}$$

whose solution is

$$y = 1, x = 0$$

which is far from the correct solution. What this example shows us is that we would like to swap rows if the entry is small and not zero. In fact, what is important is that is is small in comparison with other row entries. One can verify that if we multiply the first equation in our previous example by ϵ^{-1} we will get

$$\begin{pmatrix} 1 & \epsilon^{-1} \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \epsilon^{-1} \\ 2 \end{pmatrix}$$

and applying Gauss elimination without swapping we would run into the same issues as before. Removing the first row from the second row gives

$$\begin{pmatrix} 1 & \epsilon^{-1} \\ 0 & 1-\epsilon^{-1} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \epsilon^{-1} \\ 2-\epsilon^{-1} \end{pmatrix}$$

when ϵ is small enough again we get $fl(2-\epsilon^{-1}) = fl(1-\epsilon^{-1}) = -\epsilon^{-1}$ and so $y = 1$ and $x = 0$ will be the solution.

One method to overcome the issue we just described is, at each step j of the elimination process, swap rows so that the entry in the j, j entry is the one with maximal absolute value. With this strategy we would swap rows in the example we saw above. This strategy is called 'partial pivoting'. As it name suggests, there are more complicated pivoting approach which have been suggested, but we will not discuss here.

Lesson 11

Condition number We know that when $det(A) = 0$ there may be no solution for $Ax = b$. What happens when A is close to being singular? How do we measure this? When A is close to singlar, we will technically be able to solve $Ax = b$ but the solution will be very unstable. This is quantified using the condition number of a matrix. To do this we will need some reminders:

Definition 7. *Let V be a vector space over $F = \mathbb{R}$ or $F = \mathbb{C}$. A norm is a function $\|\cdot\| : V \to \mathbb{R}$ satisfying*

1. *$\|v\| \geq 0$ for all $v \in V$ with equality if and only if $v \neq 0$.*
2. *$\|\lambda v\| = |\lambda|\|v\|$ for all $\lambda \in F, v \in V$.*
3. *$\|v + w\| \leq \|v\| + \|w\|, \forall v, w \in V$*

Unless stated otherwise the standard norm on F^n is

$$\|x\| = \sqrt{\sum_i |x_i|^2}$$

On matrices $A \in F^{n\times n}$ we define the operator norm

$$\|A\| = \max_{x\in F^n, \|x\|=1} \|Ax\| = \max_{x\in F^n, x\neq 0} \frac{\|Ax\|}{\|x\|}$$

Lemma 4. *For every $x \in \mathbb{R}^n$ we have*

$$\|Ax\| \leq \|A\|\|x\|$$

Proof. If $x = 0$ we get equality, otherwise

$$\|Ax\| = \|x\|\frac{\|Ax\|}{\|x\|}\| \leq \|x\| \max_{y\neq 0} \frac{\|Ay\|}{\|y\|} = \|x\|\|A\|$$

□

Assume we want to solve $Ax = b$. Since we do this numerically we will only get an approximate solution $\hat{x} \approx x$, denote $\hat{b} = A\hat{x}$. We'd like to have that $\hat{b} \approx b$. Similarly, assume that we want to solve $Ax = b$, but due to numerical issues we are solving $A\hat{x} = \hat{b}$. We'd like to have that $x \approx \hat{x}$. If we measure similarity of vectors in the absolute sense, we can measure the error via the operator norms of A and A^{-1}, that is

$$\|b - \hat{b}\| = \|Ax - A\hat{x}\| \leq \|A\|\|x - \hat{x}\|$$
$$\|x - \hat{x}\| = \|A^{-1}(b - \hat{b})\| \leq \|A^{-1}\|\|b - \hat{b}\|.$$

However, in general we are interested in the relative error. This can be measure through the condition number of a matrix. The conditional number of a non-singular square matrix A is defined as

$$\kappa(A) = \|A\|\|A^{-1}\|.$$

Note that when $\kappa(A)$ is very large then A is close to being non-invertible. For example, for all $0 < \epsilon < 1$, the condition number of

$$\begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix}$$

is ϵ^{-1}. To see this note that $\|A\| = 1$ since

$$\|A\begin{pmatrix} 1 \\ 0 \end{pmatrix}\| = 1$$

and for all (x, y) with norm one, we have

$$\|A(x, y)\|^2 = x^2 + \epsilon^2 y^2 \leq x^2 + y^2 = 1.$$

A similar argument shows that $\|A^{-1}\| = \epsilon^{-1}$.

Theorem 12. *Let $A \in \mathbb{R}^{n\times n}$ be non-singular and let $x, \hat{x}, b, \hat{b}$ be non-zero vectors in $\mathbb{R}^n$ satisfying*

$$Ax = b$$

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

then

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

Proof. We have

$$\begin{aligned}\|x-\hat{x}\| &= \|A^{-1}(b-\hat{b})\| \leq \|A^{-1}\|\|b-\hat{b}\| \\ &= \|A^{-1}\|\|b-\hat{b}\|\|Ax\|\|b\|^{-1} \leq \|A^{-1}\|\|A\|\|b-\hat{b}\|\|x\|\|b\|^{-1}\end{aligned}$$

dividing by $\|x\|$ we get the inequality we wanted.

Similarly since $x = A^{-1}b$ and $\hat{x} = A^{-1}\hat{b}$ we get

$$\frac{\|b-\hat{b}\|}{\|b\|} \leq \kappa(A^{-1})\frac{\|x-\hat{x}\|}{\|x\|} = \kappa(A)\frac{\|x-\hat{x}\|}{\|x\|}$$

□

Remark: Later on we will learn about singular values of A and we will see that $\kappa(A)$ is the ratio between A's largest and smallest singular values.

Pure LU and Cholesky Decomposition In some cases we do not need any pivoting (permutation of the rows) and we have $A = LU$ for suitable lower and upper-triangular matrices.

Theorem 13. *Let $A \in \mathbb{R}^{n\times n}$ be a matrix whose principal minors $A_k = (a_{ij})_{1\leq i,j\leq k} \in \mathbb{R}^{k\times k}$ are all non-singular. Then there exists lower and upper triangular matrices L and U such that $A = LU$ and the diagonal entries of L are 1.*

Proof. We show by induction on k that $A_k = L_kU_k$ where A_k is the k-th principal minor of A and L_k, U_k are k by k lower and upper triangular matrices:

For $k = 1$ we set $L_{11} = 1, U_{11} = a_{11}$ we have $L_1U_1 = A_1$.

Assume for $k \geq 1$ we have $A_{k-1} = L_{k-1}U_{k-1}$. We are now looking for L_k, U_k satisfying $A_k = L_kU_k$. We first choose the $k-1$ principal minor of L_k, U_k to be L_{k-1}, U_{k-1} and note that we have $L_{k-1}U_{k-1} = A_{k-1}$ regardless of our choice of the remaining coordinates of L_k, U_k. We can solve for the the first $k-1$ entries of the k-th column of U by recursively solving for $i = 1, \ldots, k-1$ the equation

$$a_{ik} = \sum_{s=1}^{n} \ell_{is}u_{sk} = \sum_{s=1}^{i} \ell_{is}u_{sk}.$$

so

$$u_{ik} = a_{ik} - \sum_{s=1}^{i-1} \ell_{is}u_{sk}$$

and we are using the fact that $\ell_{ii} = 1$.

Similarly we can solve for the first $k-1$ entries of the k-th row of L via

$$a_{kj} = \sum_{\ell=1}^{n} \ell_{k\ell}u_{\ell j} = \sum_{\ell=1}^{j} \ell_{k\ell}u_{\ell j}$$

so

$$\ell_{kj} = \frac{1}{u_{jj}}\left(a_{kj} - \sum_{\ell=1}^{j-1} \ell_{kj}u_{\ell j}\right)$$

and we know that $u_{jj} \neq 0$ for all $jleqk-1$ because $A_k = L_k U_k$ is non-singular, hence U_k is non singular, and since it is upper triangular this implies that all its diagonal entries are note zero.

Finally, we set the last coordinate of L_k to be $\ell_{kk} = 1$ and set u_{kk} to be the solution of

$$a_{kk} = \sum_{\ell=1}^{k} \ell_{k\ell} u_{\ell k} = u_{kk} + \sum_{\ell=1}^{k-1} \ell_{k\ell} u_{\ell k}$$

□

Definition 8. *We say that a symmetric matrix $A = A^T \in \mathbb{R}^{n\times n}$ is positive semi-definite ($A \succeq 0$) if $\langle x, Ax\rangle \geq 0, \forall x \in \mathbb{R}^n$. If this inequality is strict for all $x \neq 0$ we say that A is positive definite ($A \succ 0$).*

Proposition 5 (Reminder)**.** *Let $A = A^T \in \mathbb{R}^{n\times n}$, then the following are equivalent*

1. *A is positive definite.*
2. *A has a basis of orthogonal eigenvectors with positive eigenvalues.*
3. *There exists a non-singular $B \in \mathbb{R}^{n\times n}$ such that $A = BB^T$.*

Proof. (1) implies (2): Since A is symmetric it has an orthogonal basis of eigenvectors $u_i, i = 1, \ldots, n$ with eigenvectors λ_i. Positive definiteness implies

$$0 < \langle u_i, Au_i\rangle = \lambda_i \|u_i\|^2$$

so λ_i is positive.

(2) implies (3): There exists a diagonal matrix $D \in \mathbb{R}^{n\times n}$ with positive entries, and an orthogonal matrix $U \in \mathbb{R}^{n\times n}$, such that

$$A = UDU^T = \left(UD^{1/2}\right)\left(UD^{1/2}\right)^T.$$

Here $D^{1/2}$ is the diagonal matrix whose diagonal entries are the square root of the diagonal entries of D.

(3) implies (1): If $A = BB^T$ for some non-singular B, then for all $v \neq 0$ we have that

$$\langle v, Av\rangle = \langle v, BB^T v\rangle = \langle B^T v, B^T v\rangle = \|B^T v\|^2 > 0$$

□

Remark If A is positive definite then it is full rank, since if x_0 is a non-zero vector in the kernel of A then we would have $\langle x_0, Ax_0\rangle = 0$. All As principal minors are also positive definite since for every $k = 1, \ldots, n$ and non-zero vector $x \in \mathbb{R}^k$ we can define $\hat{x} \in \mathbb{R}^n$ to be the vector whose first k coordinates are x and the rest are zeros and then we have that

$$\langle x, A_k x\rangle = \langle \hat{x}, A\hat{x}\rangle > 0$$

It follows from the previous theorem that A has an LU decomposition. In fact we can say more. For this we will need the following lemma

Lemma 5.
1. *Let L, M be lower triangular matrices, then LM is lower triangular.*
2. *Let L be lower triangular and non-singular, then L^{-1} is lower triangular*

Proof. The first claim is direct from the definition. We need to show that $(LM)_{ij} = 0$ if $i < j$. Indeed for all $i < j$ we have

$$(LM)_{ij} = \sum_{r=1}^{n} L_{ir} M_{rj}.$$

Note that if $r < j$ then $M_{rj} = 0$ and thus $(LM)_{ij} = 0$. Otherwise, necessarily $r > i$ and we have that $L_{ir} = 0$ and thus $(LM)_{ij} = 0$.

For the second claim, note that by Theorem 10 every lower triangular matrix L whose diagonal entries are all 1 is an inverse of products of lower triangular matrices of the form $L^{(n-1)} \ldots L^{(1)}$ where each $L^{(j)}$ is a type j matrix. By our first lemma it follows that the inverse of L is lower-triangular. If L is a general non-invertible lower triangular matrix, then all diagonal entries of L are non-zero. We define D to be the diagonal matrix with diagonal entries $D_{ii} = L_{ii}^{-1}$. Then DL is lower triangular with unit diagonal entries, and thus it has a lower triangular inverse $\bar{L}$. Multiplying by D^{-1} we obtain

$$L = D^{-1}\bar{L}$$

which is lower triangular as a product of lower triangular matrices. □

Theorem 14 (Cholesky decomposition)**.** *If $A \in \mathbb{R}^{n\times n}$ is positive definite then $A = LL^T$ for a suitable lower-triangular $L \in \mathbb{R}^{n\times n}$ with $L_{jj} > 0, j = 1, \ldots, n$. This decomposition is unique.*

Proof. By the previous theorem we know that there exist lower and upper triangular matrices L, U such that $A = LU$. Moreover L has diagonal unit entries. We have

$$LU = A = A^T = U^T L^T$$

We've seen in the previous lemma that L^{-1} is also lower-triangular. Multiplying by L^{-1} from the left and by its transpose from the right we get

$$U(L^T)^{-1} = L^{-1}U^T$$

since U and L^T are upper triangular so is $U(L^T)^{-1}$. Similarly $L^{-1}U^T$ is lower triangular and it follows that they are equal to a diagonal matrix D. This gives us

$$A = LU = LU(L^T)^{-1}L^T = LDL^T.$$

The diagonal entries of D are all positive since for $k = 1, \ldots, n$ we have for $(L^T)^{-1}e_k$ that

$$0 < \langle (L^T)^{-1}e_k, A(L^T)^{-1}e_k \rangle = D_{kk}$$

Thus we can define $D^{1/2}$ to be the diagonal matrix whose entries are the square root of D, and we then have that

$$A = LDL^T = LD^{1/2}[LD^{1/2}]^T$$

and since $LD^{1/2}$ is lower-triangular with positive diagonal entries the proof is complete. Uniqueness will be shown together with the algorithm. □

Lesson 12

Remark There exists a Cholesky decomposition also if A is not strictly positive definite, but it is not unique.

Algorithm Algorithms for Cholesky decomposition are based on Suduko like algorithms- solving equations in the correct order as we saw in the proof of Theorem 13. Here is an example:

Question 13. *Find the LL^T decomposition of the positive definite matrix*

$$A = \begin{pmatrix} 1 & 2 & 4 \\ 2 & 13 & 23 \\ 4 & 23 & 77 \end{pmatrix}$$

Answer 11. *We need to find entries for a lower-triangular L such that*

$$\begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} L_{11} & L_{21} & L_{31} \\ 0 & L_{22} & L_{32} \\ 0 & 0 & L_{33} \end{pmatrix} = \begin{pmatrix} 1 & 2 & 4 \\ 2 & 13 & 23 \\ 4 & 23 & 77 \end{pmatrix}$$

We first obtain

$$L_{11}^2 = 1$$

We know that there is a solution with positive diagonal entries so we choose $L_{11} = 1$. Next we find $L_{21} = 2$ and $L_{31} = 4$ through the equations

$$L_{21}L_{11} = 2$$

$$L_{31}L_{11} = 4.$$

Now that we know the first row and column of L, we solve for the second row and column

$$L_{21}^2 + L_{22}^2 = 13$$

so $L_{22} = 3$. Then

$$L_{31}L_{21} + L_{32}L_{22} = 23$$

implying that $L_{32} = 5$. Finally

$$\sum_{j=1}^{3} L_{3j}^2 = 77$$

so $L_{33} = 6$. Indeed we can verify that

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 1 & 2 & 4 \\ 0 & 3 & 5 \\ 0 & 0 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 4 \\ 2 & 13 & 23 \\ 4 & 23 & 77 \end{pmatrix}$$

Cholesky decomposition: Suduku algorithm In the following we describe an explicit Suduku algorihtm Assume A is strictly positive definite. We know there is a lower triangular L with positive diagonal entries such that

$$A = LL^T$$

we can now solve for the entries ℓ_{ij} of L via the equation

$$a_{ij} = \sum_{k=1}^{n} \ell_{ik}\ell_{kj}^T = \sum_{k=1}^{n} \ell_{ik}\ell_{jk} = \sum_{k=1}^{\min\{i,j\}} \ell_{ik}\ell_{jk}$$

where we used the fact that for a lower triangular matrix $L_{ik} = 0$ if $i < k$ and so we can only sum over indices k with $k \leq i, j$. We first see that '

$$a_{11} = \ell_{11}^2$$

and we choose $\ell_{11} = \sqrt{a_{11}}$. Once we solved for $\ell_{11} \neq 0$ we get ℓ_{j1} through

$$\ell_{j1} = \frac{a_{1j}}{\ell_{11}}$$

In general, once we found all the entries of the first $r-1$ columns of L, we can find the entry r,r by

$$a_{rr} = \sum_{k=1}^{r} \ell_{rk}^2$$

so

$$\ell_{rr} = \sqrt{a_{rr} - \sum_{k=1}^{r-1} \ell_{rk}^2}.$$

We know that the expression under the root is indeed positive as we proved that an LL^T decomposition exists.

We can now find the remaining values of the r column of L recursively for $i = r+1, \ldots, n$ via

$$a_{ir} = \sum_{k=1}^{r} \ell_{ik}\ell_{rk} = \ell_{ir}\ell_{rr} + \sum_{k=1}^{r-1} \ell_{ik}\ell_{rk}$$

so

$$\ell_{ir} = \ell_{rr}^{-1}(a_{ir} - \sum_{k=1}^{r-1} \ell_{ik}\ell_{rk})$$

Uniqueness and existence The equations we obtained for the elements ℓ_{ij} prove that a Cholesky decomposition, if it exists, is unique. The existence was proved earlier. It is interesting to note that although the procedure to obtain a Cholesky decomposition is very simple, the proof that it exists is relatively complex.

Complexity Computing the diagonal entry ℓ_{rr} requires $2r + O(1)$ addition, multiplication operations and one square root. Computing each of the remaining $n-r-1$ entries of the r-th column requires $2r + O(1)$ operations. So overall computing all elements in the r-th column requires $2r(n-r) + O(n)$ operations. Summing over r, we get the total number of operations we need is

$$O(n^2) + \sum_{r=1}^{n} 2r(n-r) = O(n^2) + 2n\sum_{r=1}^{n} r - 2\sum_{r=1}^{n} r^2 = O(n^2) + 2n\frac{n(n+1)}{2} - 2\frac{n(n+1)(2n+1)}{6} = \frac{n^3}{3} + O(n^2)$$

where we (again) used that fact that

$$\sum_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}.$$

This complexity is approximately half what we needed for the LU-decomposition. This makes sense: since $U = L^T$ we only need to compute L rather than L and U.

3.3 Motivation: graph Laplacian

In our discussion of Cholesky algorithm we assume that the matrix A is positive definite. Why should we encounter such a matrix in practice? A strong motivation for this is graph Laplacians which are useful for many applications, and are 'almost' positive definite as we shall soon see. Graph Laplacians are also often sparse, which will motivate the algorithms we will see later on. In the following we will define graph Laplacians and see applications for clustering problems in machine learning and numerical solution of PDEs.

Let $G = (V, E)$ be a graph, and A its adjacency matrix. That is $V = \{1, \ldots, n\}$ and E is a collection of unordered pairs of vertices $E \subseteq \{\{i, j\}|, i \neq j, i, j \in V\}$. The degree d_i of the i-th vertex is the number of its neighbors in the graph, or equivalently

$$d_i = \sum_{j=1}^{n} A_{ij}$$

The graph Laplacian is the matrix

$$L = diag(d_1, \ldots, d_n) - A$$

where $diag(d_1, \ldots, d_n)$ is the diagonal matrix whose entries are $d_1, \ldots, d_n$. Note that L is symmetric, and that it's rows sum to zero so $L\vec{1} = 0$.

example: The cycle graph $V = \{0, \ldots, n-1\}$ and $E = \{(i, i+1(modn))\}$. Each vertex has degree 2. The graph Laplacian is sparse- it has $3n$ non-zero entries.

We note that L is symmetric. In the next proposition we will see that is is positive semi-definite.

Proposition 6. *For every graph G with n vertices, and any $x \in \mathbb{R}^n$ we have*

$$\langle x, Lx \rangle = \sum_{i<j|\{i,j\}\in E} (x_i - x_j)^2 \tag{20}$$

in particular, L is positive semi-definite.

Proof.

$$\begin{aligned}
\sum_{i<j|\{i,j\}\in E} 2(x_i - x_j)^2 &= \sum_{(i,j)|\{i,j\}\in E} (x_i - x_j)^2 \\
&= \sum_{(i,j)|\{i,j\}\in E} x_i^2 - 2 \sum_{(i,j)|\{i,j\}\in E} x_i x_j + \sum_{(i,j)|\{i,j\}\in E} x_j^2 \\
&= \sum_{i=1}^{n} d_i x_i^2 - \sum_{(i,j)|\{i,j\}\in E} 2x_i x_j + \sum_{j=1}^{n} d_j x_j^2 \\
&= 2\sum_{i=1}^{n} L_{ii} x_i^2 + 2\sum_{i\neq j} L_{ij} x_i x_j = 2\langle x, Lx \rangle
\end{aligned}$$

□

We see that L is positive semi-definite. Does it have zero eigenvalues? We first note that the unit vector $\vec{1}$ always satisfies $L\vec{1} = 0$. The multiplicity of the eigenvalue 0 is related to the number of connected components of the graph which we now define.

For vertices $i, j \in V$ we say that $i \sim j$ if $i = j$, or if there is a path in the graph connecting the vertices- that is, there exist vertices $v_0 = i, v_1, \ldots, v_k = j$ such that $\{v_j, v_{j+1}\} \in E$ for all $j = 0, \ldots, k-1$. We note that $\sim$ defines an equivalence relation. The equivalence classes are called connected components of the graph.

Proposition 7. *The multiplicity of the eigenvalue* 0 *of the Laplacian of the graph G is the number of connected components of the graph.*

Proof. Let $V_1, \ldots, V_k$ be the connected components of the graph. We define corresponding vectors $x^{(1)}, \ldots, x^{(k)}$ such that $x_j^{(\ell)} = 1$ if $j \in V_\ell$ and zero otherwise. These vectors are orthogonal and it follows from (20) that they are in the kernel of L. On the other hand if x is in the kernel of L it follows from (20) that $x_i = x_j$ whenever i and j can be connected by a path in the graph, so x is in the space spanned by $x^{(1)}, \ldots, x^{(k)}$. □

Clustering : In the (binary) clustering problem we consider a graph G which is connected, but is in fact a perturbation of a graph which has two connected components. For example, there is a partition of V into V_1, V_2 and there is a very small number of edges between V_1 and V_2. The spectral clustering algorithm finds the second smallest eigevalue of L: this second eigenvalue is called the Fiedler value and the corresponding vector is the Fielder vector. When the Fiedler value $\lambda_2 = 0$ the graph is not connected. and an eigenvector which is orthogonal to the fixed eigenvector $\vec{1}$ will be positive on one connected component and negative on another. Now, if the Fiedler eigenvalue is small, we expect that partitioning according to positive and negative entries will give us a reasonably good result. This is the general ideas, there are extensions to more than two connected components and much more to say about this.

Lessons 13-14

Poisson problem In PDEs which come from physics, there almost always is a Laplacian invovled, and when this happens its discretizatoin typically involves a graph Laplacian. For example Poisson's problem on a open bounded domain $\Omega \subseteq \mathbb{R}^n$ is finding a twice differentiable function f satisfying

$$-\nabla^2 f(x) = h(x), \forall x \in \Omega$$

$$f(x) = g(x), \forall x \in \partial\Omega$$

A simple example is Poisson's problem over $\Omega = (0,1) \subseteq \mathbb{R}$ where we get

$$-f''(x) = h(x), \forall x \in (0,1) \tag{21}$$

$$f(0) = g(0), f(1) = g(1).$$

This is an equation which we can approximate by a finite dimensional linear equation as follows: for given n, we look for vectors $(f_0, \ldots, f_n)$ which will be approximate solutions in the sense that

$$f_k \approx f(k/n), k = 0, \ldots, n.$$

The values of f_0, f_n is already given to us $f_0 = g(0), f_n = g(1)$, and we use the fact that if f is twice differentiable at x then

$$-f''(x) = \frac{2f(x) - f(x+\delta) - f(x-\delta)}{\delta^2} + o(\delta^2)$$

to approximate the equation (21) via the linear equation

$$\frac{2f(\frac{k}{n}) - f(\frac{k-1}{n}) - f(\frac{k+1}{n})}{(1/n)^2} = h(\frac{k}{n})$$

or in other words

$$(2f_k - f_{k-1} - f_{k+1}) = \frac{1}{n^2} h(k/n), k = 1, \ldots, n-1.$$

To verify this technique makes sense we need to verify that (i) the linear equation has a unique solution and (ii) the vector $(f_0, \ldots, f_n)$ satisfying this equation is indeed a good approximation for the solution of the PDE, in the sense that $|f(k/n) - f_k|$ goes to zero as n goes to infinity. We will now see why (i) is true, and explain how to solve these equations in general. Question (ii) is out of scope for this course but it is worth noting at least that such results exist.

Let us analyze the linear system we obtained. We can rewrite these equations as

$$\left[\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & \ldots & 0 & 0 & 0 \\ 0 & \ldots & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & \ldots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & -1 & 2 & -1 \\ 0 & \ldots & 0 & -1 & 1 \end{pmatrix}\right] \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-2} \\ f_{n-1} \end{pmatrix} = \frac{1}{n^2} \begin{pmatrix} n^2 g(0) + h(1/n) \\ h(2/n) \\ \vdots \\ h((n-2)/n) \\ n^2 g(1) + h((n-1)/n) \end{pmatrix}$$

Let us denote the diagonal matrix on the left in the equation above by D and the equation on the right by L. Note that L is the Laplacian of the line graph from 1 to $n-1$. In particular L and D are both positive semi-definite. The matrix $L + D$ is (strictly) positive definite: Since the line graph is connected, if $x \neq \lambda \vec{1}$ for all $\lambda \in \mathbb{R}$ we have

$$\langle x, (L+D)x \rangle \geq \langle x, Lx \rangle > 0$$

On the other hand if $x = \lambda \vec{1}$ with $\lambda \neq 0$ then

$$\langle \lambda \vec{1}, (L+D)\lambda \vec{1} \rangle = \lambda^2 \langle \vec{1}, D\vec{1} \rangle = 2\lambda^2 > 0.$$

Since the matrix $L + D$ is positive definite, we can use its Cholesky decomposition to solve this linear equation approximately twice as fast as we could using LU decomposition. More significant gains can be obtained by noting that the matrix $L + D$ is sparse: that is, it has around $3n$ non-zero entries. We will now discuss how this can be useful

Sparse matrices Assume $A \in \mathbb{R}^{n\times n}$ is sparse: the number of non-zero elements $nnz(A)$ is very small compared to n^2, perhaps proportional to n. We can store A in the memory by recording the indices (i,j) for which $A_{ij} \neq 0$, and the value i,j. This is essential a $3 \times nnz(A)$ dimensional array. The number of bits needed to store a matrix in this way is $\mathcal{O}(nnz(A)(B + \log_2(n)))$, where B is the number of bits used to describe a single non-zero A_{ij}, e.g. 32 in single precision and 64 in double precision. This is much more efficient than storing the whole matrix in memory which requires $n^2 B$ bits.

Naturally, we would like to process this matrix efficiently as well. For example, we can perform matrix-vector multiplication: for full matrices the number of basic arithmetic operations used to compute Ax is $\mathcal{O}(n^2)$. For sparse matrices the number of basic arithmetic operations used is $\mathcal{O}(nnz(A))$. Similarly, we'd like to solve linear systems $Ax = b$ and compute eigenvectors of A with similar gains in efficiency. Often this is done by iterative methods which only involve matrix-vector multiplications.

In the following we will discuss the conjugate gradient method for solving sparse linear systems. To exemplify why Gauss elimination is not a great candidate for these problems, consider preforming an LU decomposition of the matrix $A \in \mathbb{R}^{n\times n}$ which has approximately $2n$ non-zero entries:

$$A = \begin{pmatrix} 2 & 2 & 2 & \dots & 2 \\ 1 & 2 & 0 & \dots & 0 \\ 1 & 0 & 2 & \dots & 0 \\ \vdots & \vdots & 0 & \ddots & 0 \\ 1 & 0 & \dots & 0 & 2 \end{pmatrix}$$

We see that after a single step of Gauss elimination we will have a full matrix on our hand.

3.4 Conjugate gradient method

A good reference for this topic is [**?**].

The conjugate gradient method aims to solve $Ax = b$ when A is positive definite (and sparse). The positive-definitess as useful as it defines an inner product:

Proposition 8. *A positive definite matrix $A \in \mathbb{R}^{n\times n}$ induces an inner product on $\mathbb{R}^n$ defined by*

$$\langle w, v\rangle_A = \langle w, Av\rangle$$

Proof. 1. Linearity: For $u, v, w \in \mathbb{R}^n$ and $t \in \mathbb{R}$ we have that

$$\langle tu + v, Aw\rangle = t\langle u, Aw\rangle + \langle v, Aw\rangle.$$

2. Symmetry: For $u, v \in \mathbb{R}^n$ we have that since A is symmetric

$$\langle u, Av\rangle = \langle A^T u, v\rangle = \langle Au, v\rangle = \langle v, Au\rangle$$

3. Positive definiteness: For $u \in \mathbb{R}^n$ we have that since A is positive definite

$$\langle u, Au\rangle \geq 0$$

and equality holds if and only if $u = 0$.

□

Notation: We denote

$$\langle u, v\rangle_A = \langle u, Av\rangle \text{ and } \|v\|_A = \sqrt{\langle v, v\rangle_A}$$

The conjugate gradient algorithm is based on a number of observations. Firstly, for a given basis $u_1, \dots, u_n$, an equation $Ax = b$ is equivalent to n different scalar equations

Lemma 6. *Let $A \in \mathbb{R}^{n\times n}, x, b \in \mathbb{R}^n$. Then $Ax = b$ if and only if*

$$\langle Ax - b, u_i\rangle = 0, \forall i = 1, \dots, n \tag{22}$$

Proof. Clearly if $Ax = b$ then (22) holds. In the other direction, assume that (22) holds. Since $u_1, \ldots, u_n$ is a basis, there exists $c_1, \ldots, c_n \in \mathbb{R}$ such that $Ax - b = \sum_{i=1}^n c_i u_i$, and so

$$\|Ax - b\|^2 = \langle Ax - b, \sum_{i=1}^n c_i u_i \rangle = \sum_{i=1}^n c_i \langle Ax - b, u_i \rangle = 0$$

□

A scalar equation $\langle Ax - b, u \rangle = 0$ clearly has many solutions x. In particular the following is a simple computations

Lemma 7. *Let $A \in \mathbb{R}^{n \times n}, x_0, b, u \in \mathbb{R}^n$ and assume that A is positive definite and $u \neq 0$. Then for*

$$t = \frac{\langle b - Ax, u \rangle}{\langle u, Au \rangle}$$

we have that

$$\langle A(x_0 + tu) - b, u \rangle = 0$$

Proof. For given A, x_0, b, u as in the lemma's statement, we want to solve the following equation in t

$$\langle A(x_0 + tu) - b, u \rangle = 0$$

which is equivalent to

$$t\langle u, Au \rangle = \langle b - Ax_0, u \rangle$$

Since A is positive definite and u is non-zero, the coefficient of t is strcitly positive, and by dividing by it we get the lemma's claim. □

Due to our previous two lemmas, a tempting idea would be to solve $Ax = b$ by iteratively solving $\langle Ax - b, u_i \rangle = 0, i = 1, \ldots, n$. The problem is that in general once solving the second equation, the first equation is no longer satisfied. This issue can be remedied, however, if the directions are A-orthogonal

Lemma 8. *If $\langle u_1, u_2 \rangle_A = 0$ and $\langle Ax - b, u_1 \rangle = 0$, then for any $t \in \mathbb{R}$,*

$$\langle A(x + tu_2) - b, u_1 \rangle = 0$$

Proof.

$$\langle A(x + tu_2) - b, u_1 \rangle = \langle Ax - b, u_1 \rangle + t\langle Au_2, u_1 \rangle = 0$$

□

The implication of this lemma is that for a given starting point x_0, when can first make a step in the direction u_1 to obtain a point x_1 solving the equation $\langle Ax_1 - b, u_1 \rangle = 0$, and then take a step in the direction u_2 to get a point x_2 solving the equation $\langle Ax_2 - b, u_2 \rangle = 0$. This point will still solve the original equation as well.

Example: Define

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \quad x_0 = (0, 0), \quad b = (1, 0)$$

The matrix A is symmetric, and it has two eigenvectors $u_1 = (1, 1)$ and $u_2 = (-1, 1)$ with eigenvalues $\lambda_1 = 3, \lambda_2 = 1$ respectively. In particular A is positive definite, and u_1, u_2 are A-orthogonal (and also orthgonal in the regular sense). It follows that we can define $x_1 = x_0 + tu_1$ such that $\langle Ax_1 - b, u_1 \rangle = 0$ according to Lemma 7, and then define $x_2 = x_1 + tu_2$ such that $\langle Ax_2 - b, u_2 \rangle = 0$ according to Lemma 7 again. According to Lemma 8 we will also have $\langle Ax_2 - b, u_1 \rangle = 0$ and thus $Ax = b$. Indeed, carrying out the calculation we see that

$$x_1 = (1/6, 1/6), x_2 = (2/3, -1/3) \text{ and } Ax_2 = b.$$

Gram-Schmidt orthogonalization Now that we see that A-orthogonality is going to play an important role, let us remind ourselves of the Gram-Schmidt procedure: If $r \in \mathbb{R}^n$ is some vector, and $u_1, \ldots, u_k \in \mathbb{R}^n$ are non-zero and pairwise A-orthogonal (not necessarily norm one), we can define

$$u = GS_A(r | \, u_1, \ldots, u_k) = r - \sum_{i=1}^k \frac{\langle r, u_i \rangle_A}{\langle u_i, u_i \rangle_A} u_i.$$

We know that $\langle u, u_i \rangle_A = 0$ for $i = 1, \ldots, k$ and that $span\{u, u_1, \ldots, u_k\} = span\{r, u_1, \ldots, u_k\}$.

Conjugate Gradient Algorithm Let $A \in \mathbb{R}^{n\times n}$ be a positive definite matrix, and $b, x_0 \in \mathbb{R}^n$. We recursively define (x_k, r_k, u_k, t_k) for $k = 1, 2, \ldots$ by

$$\begin{aligned}
&r_k = b - Ax_{k-1}\\
&u_k = GS_A(r_k|u_1, \ldots, u_{k-1}) \text{ (when } k = 1 \text{ set } u_1 = r_1)\\
&\text{if } u_k = 0 \text{ STOP!!}\\
&t_k = \frac{\langle r_k, u_k\rangle}{\|u_k\|_A^2}\\
&x_k = x_{k-1} + t_k u_k
\end{aligned}$$

Theorem 15. *The Gram-Schmidt algorithm terminates for some finite K with $1 \le K \le n+1$. For this K we have that $Ax_{K-1} = b$*

Proof. If the aglorithm was not stopped in the first K steps, then by construction $u_1, \ldots, u_{K-1}$ are all A-orthogonal. It follows that for $K = n+1$, the vectors $u_1, \ldots, u_n$ form an A-orthogonal basis for $\mathbb{R}^n$ and hence $u_{n+1} = 0$ and the algorithm will stop.

If the algorithm stops when $K = 1$ then $0 = u_1 = b - Ax_0$ so we are done. Now assume the algorithm stops after K iterations, where $2 \le K \le n+1$. We first show recursively that for all $1 \le k \le K-1$ and j with $1 \le j \le k$ we have that $\langle Ax_k - b, u_j\rangle = 0$. Indeed, for $k = 1$ we have that $\langle Ax_1 - b, u_1\rangle = 0$ by construction. Now assume that the claim is true for some $1 \le k-1 < K-1$ and all j with $1 \le j \le k-1$. Then we have by construction that $\langle Ax_k - b, u_k\rangle = 0$, and since $x_k = x_{k-1} + t_k u_k$ and $\langle u_k, u_j\rangle_A = 0$ for all $1 \le j \le k-1$, we have that

$$\langle Ax_k - b, u_j\rangle = \langle Ax_{k-1} - b, u_j\rangle = 0, j = 1, \ldots, k$$

Finally, note that once the algorithm stopped in step K, then [**Nadav:** double check, possible error]

$$0 = u_K = GS_A(r_K|u_1, \ldots, u_{K-1}) = r_K - \sum_{i=1}^{K-1} \frac{\langle r_K, u_i\rangle_A}{\langle u_i, u_i\rangle_A} u_i$$

□

What really make the conjugate gradient method useful is the ability to perform Gram-Schmidt iterations efficiently:

Proposition 9.

$$GS_A(r_k|u_1, u_2, \ldots, u_{k-1}) = GS_A(r_k|u_{k-1})$$

Proof. We first prove the following lemma

Lemma 9. *If the conjugate gradient algorithm does not stop in the first k iterations, then*

$$\mathcal{U}_k := span\{u_1, \ldots, u_k\} = span\{r_1, \ldots, r_k\} = span\{u_1, Au_1, \ldots, A^{k-1}u_1\}$$

proof of lemma. By induction. For $k = 1$ the claim is obvious. If the claim is true for $k-1$ we have for $k \ge 2$ that

$$span\{r_1, \ldots, r_k\} = span\{u_1, \ldots, u_{k-1}, r_k\} = span\{u_1, u_2, \ldots, u_k\}$$

where the first equality is by the induction hypothesis and the second is from the Gram Schmidt properties. Next note that by the induction hypothesis

$$span\{u_1, \ldots, u_{k-1}\} = span\{u_1, \ldots, A^{k-2}u_1\}.$$

It is sufficient to show that $r_k \in span\{u_1, \ldots, A^{k-1}u_1\}$. Indeed note that

$$r_k = b - Ax_{k-1} = b - A(x_{k-2} + t_{k-2}u_{k-2}) = r_{k-1} + t_{k-2}Au_{k-2} \in span\{u_1, \ldots, A^{k-1}u_1\}$$

□

The lemma shows in particular that $A\mathcal{U}_{k-2} \subseteq \mathcal{U}_{k-1}$. It follows that for $u \in \mathcal{U}_{k-2}$,

$$\langle r_k, u\rangle_A = \langle b - Ax_{k-1}, Au\rangle = 0$$

since $\langle b - Ax_{k-1}, u_i\rangle = 0$ for all $i = 1, \ldots, k-1$ and Au is spanned by $u_1, \ldots, u_{k-1}$. The claim now follows. □

Conjugate Gradient Algorithm revisited Let $A \in \mathbb{R}^{n\times n}$ be a positive definite matrix, and $b, x_0 \in \mathbb{R}^n$. We recursively define (x_k, r_k, u_k, t_k) for $k = 1, 2, \ldots$ by

$$
\begin{aligned}
&r_k = b - Ax_{k-1} \\
&u_k = GS_A(r_k|u_1, \ldots, u_{k-1}) = r_k - \frac{\langle r_k, u_{k-1}\rangle_A}{\|u_{k-1}\|_A^2} u_{k-1} \text{ (when } k = 1 \text{ set } u_1 = r_1) \\
&\text{if } u_k = 0 \text{ STOP!!} \\
&t_k = \frac{\langle r_k, u_k\rangle}{\|u_k\|_A^2} \\
&x_k = x_{k-1} + t_k u_k
\end{aligned}
$$

Remarks:

1. Since $nnz(A) \geq n$, the total complexity of each iteration is governed by the multiplication of A with vectors which requires $\mathcal{O}(nnz(A))$ operations, and so the total complexity of the algorithm is $\mathcal{O}(n \cdot nnz(A))$. Thus if A is not sparse we get $\mathcal{O}(n^3)$ but if say $nnz(A) = \mathcal{O}(n)$ then we are only $\mathcal{O}(n^2)$. We note that by assumption A is positive definite, implying all its diagonal entries are positive so $nnz(A) \geq n$.

2. In practice, the solution obtained after n iterations is not extremely exact due to numerical issues. Thus this method is more suitable to obtain a good approximate solutions for large sparse problems but is not a good replacement for Gauss elimination when searching for an exact solution of medium sized problems.

3. The conjugate gradient algorithm can be useful for other case where $Ax = b$ can be computed with less than n^2 operations. For example if $A = B^T B$, assume B is invertible and sparse. The matrix A will not necessarily be sparse, but we can still compute matrix vector multiplication $Av = BB^T v$ in $o(nnz(B))$ operations. This can be very useful, as suggested by one of the students in class in a previous lesson: assume that we have an equation $Bx = b$ where B is invertible and sparse, but is not positive definite. By multiplicating by B^T we get an equivalent equation $B^T Bx = B^T b$ where $B^T B$ is positive definite and thus we can use the conjugate gradient method on this problem.

4. Here's another example: Assume we want to find a matrix X satisfying a matrix equation such as $CXD = B$ or $C_1 XD_1 + C_2 XD_2 = B$ where $B, X, C, D, C_1, C_2, D_1, D_2 \in \mathbb{R}^{n\times n}$. Since this is ultimately a linear equation we can rewrite this equation as $Ax = b$ with $x = [X_{11}, \ldots, X_{nn}], b = [B_{11}, \ldots, B_{nn}] \in \mathbb{R}^N, N = n^2$ and some appropriate $A \in \mathbb{R}^{N\times N}$. Solving this equation with LU decomposition will have complexity of $O(N^3 = n^6)$. On the other hand, we can compute $Ax \equiv CXD$ or $Ax \equiv C_1 XD_1 + C_2 XD_2$ with complexity $O(n^3 = N^{3/2})$, and so using the conjugate gradient method we can solve this equation with complexity $O(N^{5/2} = n^5)$. Note that some matrix equations have more efficient solution. For example our first problem can be solved in $O(n^3)$ operations, when C, D are invertible. voa $X = C^{-1} BD^{-1}$.

Lessons 15-16

3.5 Computing Eigenvalues via Power method

In the following we will see a very simple method to compute the largest eigenvalue, in absolute value, of a matrix $A \in \mathbb{R}^{n\times n}$. This is also an iterative method appropriate for approximate computations of eigenvalues of large sparse matrices. To explain this simple method we start with a very simple example.

Example 2. *Consider the matrix*

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

It has two eigenvalues $\lambda_1 = -2, \lambda_2 = 1$. We note that for any $x = (x_1, x_2) \in \mathbb{R}^2$ with $x_1 = 0$ we obtain that

$$A^k x = [(-2)^k x_1, x_2]$$

We can deduce from this that, in a sense, that $A^k x$ converges to the eigenvector corresponding to λ_1. But in what sense exactly is this true?

Firstly note that in practice $\|A^k x\| \to \infty$. To resolve this issue we will consider the normalized vectors

$$x^{(k)} = \frac{1}{\|A^k x\|} A^k x.$$

These vectors have norm one. However they don't really converge, what is true is that $x^{(2k)}$ converge to $[1, 0]$ while $x^{(2k-1)}$ converge to $[-1, 0]$, these two limits are a scalar multiply of each other. What we can expect to happen is that

$$\|Ax^{(k)} - \lambda_1 x^{(k)}\| \to 0$$

Another question one could consider is how to actually find λ_1 using this method? We propose the following solution: note that if $w \in \mathbb{R}^2$ is some vector with $\langle w, v_1\rangle \neq 0$ then

$$\frac{\langle w, Av_1\rangle}{\langle w, v_1\rangle} = \lambda_1$$

Accordingly, we can try to approximate λ_1 by the ratio between $\langle w, Ax^{(k)}\rangle$ and $\langle w, x^{(k)}\rangle$.

The power sum algorithm Given $A \in \mathbb{C}^{n\times n}$ and $x_0, w \in \mathbb{C}^n$, recursively define $x^{(k)}, \lambda^{(k)}$ via

$$x^{(k)} = \frac{Ax^{(k-1)}}{\|x^{(k-1)}\|}$$
$$\lambda^{(k)} = \frac{\langle w, Ax^{(k)}\rangle}{\langle w, x^{(k)}\rangle}$$

This procedure converges to the 'first' eigenvector, under some conditions

Theorem 16. *Let $n \geq 2$ be an integer, $A \in \mathbb{C}^{n\times n}$ and $x^{(0)}, w \in \mathbb{C}^n$, and assume that*

1. *A has a basis of eigenvectors $u_1, \ldots, u_n \in \mathbb{C}^n$, corresponding to eigenvalues $\lambda_1, \ldots, \lambda_n \in \mathbb{C}$.*
2. *$|\lambda_1| > |\lambda_k|, \forall k = 2, \ldots, n$.*
3. *When $x^{(0)}$ is expressed as a linear combination*

$$x^{(0)} = a_1 u_1 + a_2 u_2 + \ldots + a_n u_n$$

   The coefficient a_1 is not zero.

4. *$\langle w, u_1\rangle \neq 0$*

Then $x^{(k)}$ is well defined for all k, $\lambda^{(k)}$ is well defined for all large enough k, and

$$\lambda^{(k)} \to \lambda_1, \quad \|Ax^{(k)} - \lambda_1 x^{(k)}\| \to 0$$

Remark: Of the conditions above the most restrictive is the second one: the power sum algorithm may not converge if this condition is not satisfied, and if the first two eigenvalues have similar magnitude then convergence will be very slow. The other conditions are less restrictive: there are many classes of matrices for which an eigenbasis exists, as we know. For a given A, if we randomly choose complex $x^{(0)}$ and w, say according to the uniform distribution on the unit $2n$-dimensional cube, the probability of obtaining 'bad' $x^{(0)}$ and w is zero.

Proof. We first prove recursively that for all $k \geq 1$ we have that $x^{(k)}$ is well defined and equal to

$$x^{(k)} = \frac{A^k x^{(0)}}{\|A^k x^{(0)}\|} \tag{23}$$

Note that for all k we have that

$$A^k x^{(0)} = \lambda_1^k a_1 u_1 + \lambda_2^k a_2 u_2 + \ldots \lambda_n^k a_n u_n \tag{24}$$

$$= a_1 \lambda_1^k \left(u_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k \frac{a_2}{a_1} u_2 + \ldots + \left(\frac{\lambda_n}{\lambda_1}\right)^k \frac{a_n}{a_1} u_2 \right) \tag{25}$$

By assumption $\lambda_1, a_1 \neq 0$ and so the vector we obtained is non-zero. For $k = 1$ we have that $x^{(1)}$ is well defined and satisfies (23). Now assume that for some k we know that (23) holds, then for $k+1$ we get that

$$x^{(k+1)} = \frac{Ax^{(k)}}{\|Ax^{(k)}\|} = \frac{\|A^k x^{(0)}\|}{\|A^k x^{(0)}\|} \frac{A^{k+1} x^{(0)}}{\|A^{k+1} x^{(0)}\|} = \frac{A^{k+1} x^{(0)}}{\|A^{k+1} x^{(0)}\|}$$

Next we note that the expression (25) we obtained above can be rewritten as

$$A^k x^{(0)} = a_1 \lambda_1^k \left(u_1 + \epsilon_k\right)$$

where ϵ_k is a vector which converges to zero as $k \to \infty$. We then have that

$$\lambda^{(k)} = \frac{\langle w, Ax^{(k)} \rangle}{\langle w, x^{(k)} \rangle} = \frac{\langle w, A^{k+1} x^{(0)} \rangle}{\langle w, A^k x^{(0)} \rangle} = \frac{a_1 \lambda_1^{k+1} \langle w, u_1 + \epsilon_{k+1} \rangle}{a_1 \lambda_1^k \langle w, u_1 + \epsilon_k \rangle} \to \lambda_1$$

where we used the fact that the limit in the denominator is not zero. Similarly we have that

$$\|Ax^{(k)} - \lambda_1 x^{(k)}\| = \frac{1}{\|A^k x^{(0)}\|} \|a_1 \lambda_1^k A\left(u_1 + \epsilon_k\right) - \lambda_1 a_1 \lambda_1^k \left(u_1 + \epsilon_k\right)\| = \frac{\|A\epsilon_k - \lambda_1 \epsilon_k\|}{\|u_1 + \epsilon_k\|} \to 0$$

□

Complexity per step Every step in the algorithm requires a matrix-vector multiplication, and at least n operations, e.g. to compute the norm, so the complexity of each step is $\mathcal{O}(n + nnz(A))$. This needs to be multiplied by the number of steps which is determined by the accuracy we are looking for. **Finding the largest and smallest eigenvalue of a symmetric matrix** Assume $A \in \mathbb{R}^{n \times n}$ is a symmetric matrix with $\lambda_n < \lambda_{n-1} \leq \ldots \leq \lambda_2 < \lambda_1$, and assume $|\lambda_1| > |\lambda_n|$. We can find λ_1 using the power method. To find λ_n, we can look at $\tilde{A} = \lambda_1 I_n - A$ whose eigenvalues are

$$\tilde{\lambda}_j = \lambda_1 - \lambda_j.$$

These are now all non-negative and $\tilde{\lambda}_n > \tilde{\lambda}_j, \forall j \neq n$. Thus we can apply the power method to $\tilde{A}$ and find $\tilde{\lambda}_n$, and then find λ_n via $\lambda_n = \lambda_1 - \tilde{\lambda}_n$.

Finding the first and second largest magnitude eigenvalues Say we have some symmetric real matrix A with eigenvalues $\lambda_1, \ldots, \lambda_n$ satisfying $|\lambda_1| > |\lambda_2| > |\lambda_3| \geq |\lambda_4| \geq \ldots \geq |\lambda_n|$. Let u_i be an orthonormal basis such that $Au_i = \lambda_i u_i$.We want to find λ_1 and λ_2. We can find λ_1 using the power method. To find λ_2, we define a matrix Δ on the eigenbasis by $\Delta u_1 = u_1$ and $\Delta u_j = 0, \forall j > 1$. We then have that the eigenvalues of $\tilde{A} = A - \lambda_1 \Delta$ are $\tilde{\lambda}_1 = 0$ and $\tilde{\lambda}_j = \lambda_j, \forall j = 2, \ldots, n$. The largest magnitude eigenvalue of $\tilde{A}$ is $\tilde{\lambda}_2 = \lambda_2$ and so we can find it by using the power method. In theory, if all eigenvalues have distinct magnitue we can find all of them using this procedure recursively. In practice this is not usually recommended if you want to find many eigenvalues. For example, Matlab has two functions to compute eigenvalues: 'eig' which is recommended in general, and gives a full eigenvalues decompositions, and 'eigs', which is recommended for finding a small number of eigenvalues, for sparse matrices.

Remark: The matrix Δ is the tensor product of the vector u_1 with itself. To define tensor products it is useful to use a common notation in computer science, where vectors $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ are identified with a degenerate 'column matrix' $x = (x_{11}, \ldots, x_{n1}) \in \mathbb{R}^{n\times 1}$. This is usually called a column vector. If one takes the transpose of a column vector we get a 'row vector' $x^T \in \mathbb{R}^{1\times n}$. The inner product of two vectors is the same as matrix multiplication, that is if $x = (x_{11}, \ldots, x_{n1}) \in \mathbb{R}^{n\times 1}$ and $y = (y_{11}, \ldots, y_{n1}) \in \mathbb{R}^{n\times 1}$ then $x^T y$ is a 1×1 matrix

$$\left(x^T y\right)_{11} = \sum_{j=1}^{n} x_{1j}^T y_{j1} = \sum_{j=1}^{n} x_{ji} y_{j1} = \langle x, y\rangle.$$

Similarly, one could look at $xy^T \in \mathbb{R}^{n\times n}$. This matrix is called the tensor product of the vectors, and

$$(xy^T)_{ij} = \sum_{k=1}^{1} x_{ik} y_{jk} = x_{i1} y_{j1}.$$

Note that if u_1 and u_2 are orthogonal then by the accosiativity of matrix multiplication

$$(u_1 u_1^T)u_1 = u_1(u_1^T u_1) = \|u_1\|^2 u_1$$

and

$$(u_1 u_1^T)u_2 = u_1(u_1^T u_2) = \langle u_1, u_2\rangle u_1 = 0$$

Remark: finding the second smallest value of a Laplacian We saw that the Laplacian L of a connected graph is a symmetric matrix with eigenvalues $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n$, and the first eigenvector is $\vec{1}$. We saw that for 'weakly connected' graphs, finding the second eigenvalue of the Laplacian can be useful for the clustering problem. Let us see how this can be done using the power method: For this we will assume that $\lambda_2 > 0$ and $\lambda_n > \lambda_{n-1}$. Applying the power method to L (with an appropriate choice of $w, x^{(0)}$ will give us λ_n. To get λ_2 we can use the fact that the matrix $J_n = \frac{1}{n}\vec{1}\,\vec{1}^T$ satisfies $J_n\vec{1} = \vec{1}$ and $J_n v = 0$ for all v orthogonal to $\vec{1}$. We deduced that the eigenvalues of $\tilde{L} = \lambda_n(I_n - J_n) - L$ are

$$\tilde{\lambda_1} = 0, \tilde{\lambda_2} = \lambda_n - \lambda_2, \ldots \tilde{\lambda_n} = \lambda_n - \lambda_n = 0$$

We now have that $\tilde{\lambda_2} > \tilde{\lambda_j} \geq 0, \forall j \neq 2$, and we can find $\tilde{\lambda_2}$ by applying the power method and λ_2 via $\tilde{\lambda_2} = \lambda_n - \lambda_2$.

Inverse power method If we wish to compute the smallest eigenvalue in magnitude $|\lambda_n| < |\lambda_j|, j = 1, \ldots, n-1$ we can do this by applying the power method to A^{-1} since λ_n^{-1} is the largest eigenvalue of A^{-1}. However instead of computing

$$x_{k+1} = A^{-1} x_k$$

it is preferrable to compute x_{k+1} as the solution to the linear equation

$$Ax_{k+1} = x_k$$

This can be done by computing the LU decomposition of A once in $O(n^3)$ and then solving $P^T LU x_{k+1} = x_k$ in $O(n^2)$ per iteration. This requires more operations than the regular power method.

3.6 Singular Value Decomposition

The singular value decomposition (SVD) is a very useful matrix decomposition which is applicable *for all* matrices (including non-square matrices). Let us begin with some preliminaries.

First let us remind ourselves that a matrix $U \in \mathbb{R}^{n\times n}$ is *orthogonal* if $UU^T = I_n$, where $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix. If U is an orthogonal matrix then

1. $U^T = U^{-1}$
2. $\|Ux\| = \|x\|$ and $\langle Ux, Uy\rangle = \langle x, y\rangle$, for all $x, y \in \mathbb{R}^n$.

A $m \times n$ matrix D is *diagonal* if $D_{ij} = 0$ whenever $i \neq j$.

Theorem 17 (Singular value decompostion)**.** *Let $m, n \in \mathbb{N}$ and denote $\nu = \min\{n, m\}$. For every $A \in \mathbb{R}^{m\times n}$, there exists an $m \times m$ orthogonal matrix U, a diagonal $m \times n$ matrix D with non-negative entries $D_{ii} = \sigma_i$ satisfying*

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\nu \geq 0$$

and an orthogonal $n \times n$ matrix V, such that

$$A = UDV^T.$$

Remark: The decomposition in the theorem is called a singular value decomposition (SVD). The diagonal elements σ_i are called the matrices singular values.

Question 14. *Using what you know on the unitary eigendecomposition of square matrices, find a singular value decompositions $A = UDV^T$ for the following matrices*

1. *$A \in \mathbb{R}^{n\times n}$ a positive semi-definite matrix.*
2. *$A \in \mathbb{R}^{n\times n}$ a symmetric matrix.*
3. *$A = U \in \mathbb{R}^{n\times n}$ an orthogonal matrix.*

Proof. Assume without loss of generality that $n \leq m$ (otherwise, we can look at A^T, obtain a SVD decomposition for it, and then transopose to get an SVD decomposition for A). We need to show that there exists U, V, D such that

$$U^T AV = D$$

or in other words, we need to find n orthonormal vectors $v_1, \ldots, v_n$ in $\mathbb{R}^n$ (the columns of V), m orthonormal vectors in $\mathbb{R}^m$ (the columns of U), and a digonal matrix $D \in \mathbb{R}^{m\times n}$ with values $D_{ii} = \sigma_i$ satisfying $\sigma_1 \geq \ldots \geq \sigma_n \geq 0$, such that

$$\langle u_i, Av_j\rangle = D_{ij} = \begin{cases} \sigma_i & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{26}$$

We choose $v_1, \ldots, v_n$ to be an orthonormal eigenbasis to the positive semi-definite matrix $A^T A$, which has non-negative eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$, and set $\sigma_i = \sqrt{\lambda_i}$. Let r be the last index for which $\lambda_r > 0$. We now choose $u_1, \ldots, u_r$ to be

$$u_i = \sigma_i^{-1} Av_i, i = 1, \ldots, r$$

note that these form an orthonormal family as for $1 \leq i \leq j \leq r$ we have

$$\langle u_i, u_j\rangle = \sigma_i^{-1}\sigma_j^{-1}\langle Av_i, Av_j\rangle = \sigma_i^{-1}\sigma_j^{-1}\langle v_i, A^T Av_j\rangle = \sigma_i^{-1}\sigma_j\langle v_i, v_j\rangle = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}.$$

If $r < m$, we choose $u_{r+1}, \ldots, u_m$ by arbitrarily completing $u_1, \ldots, u_r$ to an orthonormal basis for $\mathbb{R}^m$. We now have that if $j \leq r$ then

$$\langle u_i, Av_j\rangle = \langle u_i, \sigma_j u_j\rangle = \begin{cases} 0 & \text{if } i \neq j \\ \sigma_j & \text{if } i = j \end{cases}$$

and if $j > r$ then

$$\|Av_j\|^2 = \langle Av_j, Av_j\rangle = \langle v_j, A^T Av_j\rangle = \lambda_j = 0$$

and therefore

$$\langle u_i, Av_j\rangle = \langle u_i, \vec{0}\rangle = \begin{cases} 0 & \text{if } i \neq j \\ 0 = \sigma_j & \text{if } i = j \end{cases}$$

So we get (26) and we are done. □

Proposition 10. *Let $A \in \mathbb{R}^{m\times n}$ be a matrix with singular value decompositions*

$$A = U_1 D_1 V_1^T = U_2 D_2 V_2^T,$$

then $D_1 = D_2$.

Proof. We prove for the case $m = n$ only. Note that

$$A^T A = U_1 D_1^T D_1 U_1^T = V_2 D_2^T D_2 V_2^T$$

and $D_i^T D_i = D_i^2, i = 1, 2$ is an $n \times n$ diagonal matrix. Uniqueness of the eigendecomposition implies that $D_1^2 = D_2^2$, and for diagonal matrices with non-negative entries this implies that $D_1 = D_2$. □

3.7 SVD applications

Matrix norm and condition number

Proposition 11. *If A is a $m \times n$ matrix with singular values $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_\nu$, where $\nu = \min\{n, m\}\}$, then $\|A\| = \sigma_1$*

Proof. We prove for the case $m = n$ only.

Let us first compute the operator norm of D. Note that for all $v \in \mathbb{R}^n$ with norm one we have

$$\|Dv\|^2 = \sum_{i=1}^{n} (D_{ii} v_i)^2 \leq \sigma_1^2 \sum_{i=1}^{n} v_i^2 = \sigma_1^2$$

so $\|D\| \leq \sigma_1$. On the other hand we have that $\|De_1\| = \|\sigma_1 e_1\| = \sigma_1$, and thus $\|D\| = \sigma_1$.

Now if $A = UDV^T$ we get that for all q with norm one,

$$\|Aq\| = \|UDV^T q\| = \|DV^T q\| \leq \|D\|\|V^T q\| = \|D\| = \sigma_1,$$

and thus

$$\|A\| = \max_{q \in \mathbb{R}^n, \|q\|=1} \|Aq\| \leq \sigma_1$$

on the one hand, since $D = U^T AV$ we can reverse this argument and obtain that

$$\sigma_1 = \|D\| \leq \|A\|,$$

and thus $\|A\| = \sigma_1$. □

Remark: If A is a square, invertible matrix, then its SVD decomposition consists of square matrices $A = UDV^T$ and $det(D) \neq 0$ so all diaongal elements are non zero, and we get

$$A^{-1} = VD^{-1}U^T$$

in particular we get that

$$\|A^{-1}\| = \sigma_n^{-1}$$

and so the condition number of A is

$$\kappa(A) = \|A\|\|A^{-1}\| = \frac{\sigma_1}{\sigma_n}$$

Low rank matrices, compression, and dimensionality reduction We now consider the SVD decomposition of low rank matrices, and consider the applications of this for two applications: matrix compression and dimensionality reduction:

Assume that $X \in \mathbb{R}^{m \times n}$, and that X has rank $r < \min\{m, n\}$. Note that this means that $\sigma_r > 0$ and $\sigma_{r+1} = 0$ We show that the SVD decomposition of such a matrix can be stored more efficiently than the matrix X itself, and moreover, that there is a matrix $Y \in \mathbb{R}^{r \times n}$ such that the distances between the columns in X are the same as the distances of the columns in Y.

Proposition 12. *Let $X \in \mathbb{R}^{m \times n}$ be a matrix with rank $r \leq \min\{m, n\}$, then there exists a diagonal matrix $D \in \mathbb{R}^{r \times r}$ with positive diagonal entries, and matrices $U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}$ whose columns form an orthonormal system, such that*

$$X = UDV^T.$$

Moreover, denoting $Y = DV^T$, and denoting the i-th columns of Y and X by $Y_{*,i}$ and $X_{*,i}$ respectively, we have that

$$\|Y_{*,i} - Y_{*,j}\| = \|X_{*,i} - X_{*,j}\|, \forall 1 \leq i \leq j \leq n.$$

Proof. The SVD decomposition of the matrix X is

$$X = UDV^T$$

where $U \in \mathbb{R}^{m \times m}, D \in \mathbb{R}^{m \times n}, V \in \mathbb{R}^{n \times n}$ and $D_{jj} = 0$ for $j > r$. We rewrite these matrices as block matrices

$$U = [U_1, U_2], V = [V_1, V_2], \text{ and } D = \begin{pmatrix} D_1 & \\ & 0 \end{pmatrix}$$

where $D_1 \in \mathbb{R}^{r\times r}$, and $U_1 \in \mathbb{R}^{m\times r}$ and $V_1 \in \mathbb{R}^{n\times r}$ are matrices whose columns form an orthonormal system, as stated in the proposition's statement. We then have

$$X = [U_1, U_2]\begin{pmatrix} D_1 & \\ & 0 \end{pmatrix}\begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} = [U_1, U_2]\begin{pmatrix} D_1 V_1^T \\ \vec{0} \end{pmatrix} = U_1 D_1 V_1^T.$$

This proves the first part of the proposition. Next, denoting $Y = D_1 V_1^T \in \mathbb{R}^{r\times n}$, and the columns of Y and X as described in the proposition's statement, we have from the previous equation that

$$X = [U_1, U_2]\begin{pmatrix} D_1 V_1^T \\ \vec{0} \end{pmatrix} = U\begin{pmatrix} Y \\ \vec{0} \end{pmatrix} = U\tilde{Y}$$

where $\tilde{Y}$ is the matrix obtained by appending the 0 matrix to Y in the equation above. Thus we have that $X_{*j} = U\tilde{Y}_{*j}$ for all $j = 1\ldots, n$, and thus

$$\|X_{*i} - X_{*j}\| = \|U\tilde{Y}_{*i} - U\tilde{Y}_{*j}\| = \|\tilde{Y}_{*i} - \tilde{Y}_{*j}\| = \|Y_{*i} - Y_{*j}\|.$$

□

Applications We consider two applications of the last propositions:

1. **Matrix compression:** Note that while storing X directly requires nm numbers, we can alternatively store $U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{n\times r}$ using mr and nr numbers respectively, and the diagonal matrix D using r numbers. Altogether we need $(n + m + 1) \times r$ numbers. This can be much less than the nm numbers required in the standard method, if $r \ll \min\{n, m\}$.

   In class we will see an example for image compression which can be found here:

   https://www.mathworks.com/help/matlab/math/image-compression-with-low-rank-svd.html

2. **Dimensionality reduction** As we saw, if $X \in \mathbb{R}^{m\times n}$ is rank r, we can find $Y \in \mathbb{R}^{r\times n}$ such that this pairwise distances between the columns of X and the columns of Y are identical. Thus if we only care about pairwise distances we get even better compression. In class we will see applications of this for data visualization (where $r = 2$ or $r = 3$).

SVD Truncation Often the rank of the matrix $X \in \mathbb{R}^{m\times n}$ is not very small, but it is possible to find a small r such that $\sigma_{r+1}, \ldots$ is very small relatively to σ_1. In this case a reasonable strategy is to use truncation: that is, if

$$X = UDV^T$$

we can consider, for every r, replacing

$$D = diag(\sigma_1, \ldots, \sigma_r, \sigma_{r+1}, \ldots, \sigma_\nu) \text{ with } D^{(r)} = diag(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$$

(where $\nu = \min\{m, n\}$), and $X = UDV^T$ with $X^{(r)} := UD^{(r)}V^T$. The matrix $X^{(r)}$ can then be compressed (or dimensionality reduction can be applied). The smaller r is taken, the efficiency of the compression is improved while the accuracy is reduced.

The following gives a theoretical justification for this procedure, by showing that SVD truncation is the best rank r approximation of a given matrix, when using the operator norm (this is also true for the Frobenious norm, but we will not show this)

Theorem 18. *Let $X \in \mathbb{R}^{m\times n}$, let $r \le \min\{m, n\}$ and let $X^{(r)} \in \mathbb{R}^{m\times n}$ be the matrix obtained from X by SVD-truncation. Then for all rank-r matrices $B \in \mathbb{R}^{m\times n}$*

$$\|X - X^{(r)}\| \le \|X - B\|$$

(the norm here is the operator norm).

Proof. Recall that if $X = UDV^T$ then we define $X^{(r)} := UD^{(r)}V^T$. We first compute

$$\|X - X^{(r)}\| = \|U(D - D^{(r)})V^T\| = \|Udiag(0, \ldots, 0, \sigma_{r+1}, \ldots, \sigma_n)V\| = \sigma_{r+1},$$

where we used the fact that the norm of a matrix is it's largest singular value. Now let B be some rank r matrix. It remains to show that $\|X - B\|$ will not be smaller than σ_{r+1}. To see this, we will consider vectors in the $r+1$ dimensional vector space

$$W_{r+1} = \{q \in \mathbb{R}^n | q_{r+2}, \ldots, q_n = 0\}.$$

Note that if q has unit norm, then so does Vq, and

$$\|XVq\|^2 = \|(UDV^T)Vq\|^2 = \|UDq\|^2 = \|Dq\|^2 = \sum_{i=1}^{r+1} \sigma_i^2 q_i^2 \geq \sigma_{r+1}^2 \sum_{i=1}^{r+1} q_i^2 = \sigma_{r+1}^2.$$

On the other hand, since B has rank r, so does BV, it follows that the restriction of BV to the $r+1$ dimensional W_{r+1} is not injective, and thus there exists some $q \in W_{r+1}$ with $BVq = 0$ and we can normalize q so that is has unit norm. We then obtain

$$\|X - B\| = \|X - B\|\|Vq\| \geq \|(X - B)Vq\| = \|XVq\| \geq \sigma_{r+1}$$

as required. □

Projection onto orthogonal matrices Recall the Frobeniuos inner product

$$\langle A, B \rangle_F = \sum_{ij} A_{ij} B_{ij} = trace(AB^T)$$

Proposition 13. *Let $A \in \mathbb{R}^{n \times n}$ with SVD*

$$A = P\Sigma Q^T$$

then the orthogonal matrix PQ^T satisfies that for any other orthogonal matrix U we have that

$$\|A - PQ^T\|_F \leq \|A - U\|_F$$

Proof. Note that for any U

$$\|A - U\|_F^2 = \|A\|_F^2 + \|U\|_F^2 - 2\langle A, U \rangle_F$$

and since always $\|U\|_F^2 = n$ we need to maximize

$$\langle A, U \rangle_F = trace(P\Sigma Q^T U^T) = trace(\Sigma Q^T U^T P)$$

Taking $U = PQ^T$ we get

$$trace(\Sigma Q^T U^T P) = trace(\Sigma) = \sum_i \sigma_i$$

on the other hand for every U we have that $\tilde{U} = Q^T U^T P$ is also orthogonal, in particular $|\tilde{U}_{ii}| \leq 1$, and so

$$trace(\Sigma Q^T U^T P) = trace(\Sigma \tilde{U}^T) = \sum_i \sigma_i U_{ii} \leq \sum_i \sigma_i$$

□

Lesson 17

4 Approximation and interpolation

In this chapter we consider two related problems:

1. Approximation: Given some function $f : \mathbb{R} \to \mathbb{R}$ how do we approximate it with a finite dimensional basis of functions. e.g,. we can approximate $f(x) = \sin(x)$ by its Taylor approximation to some order. Can we do better?

2. Interpolation: given $n+1$ samples of a function

$$x_i, y_i = f(x_i), i = 0, \ldots, n$$

how to find a function h with

$$h(x_i) = y_i$$

For example, when plotting graphs we typically use some interpolation scheme.

4.1 Polynomial interpolation

We consider the following problem. We are give $n+1$ *distinct* points $x_0, x_1, \ldots, x_n \in \mathbb{R}$ and values $y_0, \ldots, y_n \in \mathbb{R}$. We want to find a polynomial p of lowest possible degree for which

$$p(x_i) = y_i, i = 1, \ldots, n$$

such a polynomial is said to interpolate the data. We denote the space of polynomials of degree $\leq n$ by $\mathbb{R}_n[x]$. We prove

Theorem 19. *For any $n+1$* distinct *points $x_0, x_1, \ldots, x_n \in \mathbb{R}$ and values $y_0, \ldots, y_n \in \mathbb{R}$ there exists a unique polynomial $p \in \mathbb{R}_n[x]$ such that*

$$p(x_i) = y_i, i = 0, 1 \ldots, n$$

We will see two proofs for this theorem:

Proof. A polynomial

$$p(x) = \sum_{k=0}^{n} c_k x^k$$

satisfies

$$\begin{aligned} p(x_0) &= y_0 \\ p(x_1) &= y_1 \\ &\ldots \\ p(x_n) &= y_n \end{aligned}$$

if and only if $c_0, \ldots, c_n$ solve the linear equations (in this example $n = 3$)

$$\begin{pmatrix} 1 & x_0 & x_0^2 & x_0^3 \\ 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

The matrix on the left hand side is called the Van-der-Monde matrix. It is non-singular. Therefore this equation has a unique solution and thus there is a unique interpolation polynomial in $\mathbb{R}_n[x]$.

We will now see a second proof which does not assume the Vandermonde matrix is non-singular (but implies that it is, since it shows there is a unique solution). Recall that in an equation of the form $Ax = b$ where A is square, there is a unique solution for every b, if and only if the homogenous equation $Ax = 0$ has the unique solution $x = 0$. This corresponds to proving the following lemma

Lemma 10. *For every $n \in \mathbb{N}$ and distinct points $x_0, \ldots, x_n \in \mathbb{R}$, the only polynomial $p \in \mathbb{R}_n[x]$ satifying $p(x_i) = 0$ for all $i = 0, \ldots, n$ is the zero polynomial.*

Proof. By induction: if $n = 1$ and $p \in \mathbb{R}_1[x]$ is zero on two points then $p = 0$ everywhere. Now for $n \geq 2$, assume correctness for $n-1$, and us prove for n. If $p \in \mathbb{R}_n[x]$ is zero on $n+1$ different points, then by Rolle's theorem, for every $i = 0, \ldots n-1$ there exists some $y_i \in (x_i, x_{i+1})$ such that $p'(y_i) = 0$. The derivative polynomial is in $\mathbb{R}_{n-1}[x]$ and is zero on n points, hence $p' = 0$ everywhere by the induction hypothesis. It follows that $p \in \mathbb{R}_1[x]$ and it is zero on $n + 1 > 2$ points, and therefore $p = 0$ again by the induction hypothesis. □

□

Interpolation polynomial in Newton's form To actually compute the interpolation polynomial, one could solve the linear equation above which involves the Vandermonde matrix. However this is a full matrix with no special structure, so this is relatively ineffecient, requiring $O(n^3)$ operations. In addition, this matrix is known to often have a bad condition number...

In the Newton form of the interpolation polynomial, we write the inerpolation polynomial as a linear combination of the functions

$$\{1, x - x_0, (x - x_0)(x - x_1), \ldots, (x - x_0)(x - x_1) \ldots (x - x_{n-1})\}$$

that is:

$$p_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots + c_n(x - x_0)(x - x_1) \ldots (x - x_{n-1})$$

The coefficients c_i of the interpolation polynomial can be obtained through via the equations

$$\begin{aligned} p_n(x_0) &= y_0 \\ p_n(x_1) &= y_1 \\ &\ldots \\ p_n(x_n) &= y_n \end{aligned}$$

which are linear equations which can be written in matrix form as (in this example $n = 3$)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & x_1 - x_0 & 0 & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) & 0 \\ 1 & x_3 - x_0 & (x_3 - x_0)(x_3 - x_1) & (x_3 - x_0)(x_3 - x_1)(x_3 - x_2) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

We get a matrix equation of the form $Lc = y$, where the matrix L can be computed in $O(n^2)$ steps since we can compute every row of L in $O(n)$, and we can then solve the equation in $O(n^2)$ since L is lower triangular. Note that since all the x_i are distinct the diagonal entries of L are non-zero. This gives another proof to the existence of an interpolation polynomial (uniqueness would require some additional work)

Example Consider the function

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

and assume we are given samples

$$x_0 = -1, y_0 = f(x_0) = 3$$

$$x_1 = 0, y_1 = f(x_1) = 1$$

$$x_2 = 1, y_2 = f(x_2) = 1$$

We will find the Newton form of the interpolation polynomial p_2. Due to uniqueness of the interpolation polynomial and since f is a degree 2 polynomial we expect to get $p_2 = f$. We are looking for coefficeints c_0, c_1, c_2 such that

$$c_0 + c_1(x_i + 1) + c_2(x_i + 1)(x_i) = y_i, i = 0, 1, 2.$$

We obtain the linear equation

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix}$$

Solving these equations from top to bottom we obtain

$$c_0 = 3, c_1 = -2, c_2 = 1$$

all in all we get

$$p_2(x) = 3 - 2(x+1) + x(x+1)$$

which is the same as the original f since

$$p_2(x) = 3 - 2(x+1) + x(x+1) = x^2 - x + 1$$

We next see another useful basis for the interpolation polynomial:

Lagrange form of the interpolation polynomial Consider the interpolation problem defined by $(x_i, y_i), i = 0, \ldots, n$ in the special case where $y_0 = 1$ and $y_1 = y_2 = \ldots = y_n = 0$. In this case we can interpolate the data by the degree n polynomial

$$\ell_0(x) = \prod_{i=1}^{n} \frac{x - x_i}{x_0 - x_i}\,.$$

similarly, when $y_j = 1$ and all other $y_i = 0, i \neq j$ we can interpolate the data by

$$\ell_j(x) = \prod_{i:\, i \neq j} \frac{x - x_i}{x_j - x_i}.$$

This gives us a basis of degree n polynomials $\ell_0, \ldots \ell_n$ such that

$$\ell_i(x_j) = \delta_{ij}$$

Given general points $(x_i, y_i), i = 0, \ldots, n$ to interpolate we can define an interpolation via

$$p_n(x) = \sum_{j=0}^{n} y_j \ell_j(x).$$

and then for every $i = 0, \ldots, n$ we have that

$$p_n(x_i) = \sum_{j=0}^{n} y_j \ell_j(x) = y_i.$$

Example We again consider the function

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

and assume we are given samples

$$x_0 = -1, y_0 = f(x_0) = 3$$
$$x_1 = 0, y_1 = f(x_1) = 1$$
$$x_2 = 1, y_2 = f(x_2) = 1$$

We will find the Lagrange form of the interpolation polynomial p_2. We have

$$\ell_0(x) = \prod_{i=1}^{2} \frac{x - x_i}{x_0 - x_i} = \frac{x(x-1)}{(-1-0)(-1-1)} = \frac{x^2 - x}{2}$$

and

$$\ell_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{(x-1)(x+1)}{(0-(-1))(0-1)} = -x^2 + 1$$

and

$$\ell_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{(x+1)x}{2 \cdot 1} = \frac{x^2 + x}{2}$$

we then get

$$p_2(x) = y_0\ell_0(x) + y_1\ell_1(x) + y_2\ell_2(x) = \frac{3}{2}(x^2 - x) - x^2 + 1 + \frac{1}{2}(x^2 + x)$$

which in the standard basis is indeed

$$f(x) = x^2 - x + 1.$$

Lagrange vs. Newton, what is better? This depends on what we are trying to do. In general, an advantage of the Lagrange form is that once it is computed, changing y values becomes very easy. On the other hand, Newton's form is useful if you want to gradually add more interpolation points since it recursively builds the interpolants: If p_n of degree $\leq n$ satisfies $p_n(x_i) = y_i$ for $i = 0, \ldots, n$, and we want to build a polynomial p_{n+1} of degree $n+1$ which satisfies these equations and also satisfies $p_{n+1}(x_{n+1}) = y_{n+1}$, then using the Newton form, we write p_{n+1} as

$$p_{n+1}(x) = p_n(x) + c_{n+1} \prod_{i=0}^{n} (x - x_i),$$

where c_{n+1} is defined so that $p_{n+1}(x_{n+1}) = y_{n+1}$.

4.2 Approximation via interpolation

Given a continuous function f on $[a, b]$, how well can it be approximated by interpolation polynomials?

Theorem 20. *Let f be $n+1$ times continuously differentiable in an open interval containing $[a, b]$, and let p_n be the degree $\leq n$ interpolation polynomial of f at $n+1$ distinct points $x_0, x_1, \ldots, x_n \in [a, b]$. Then for every $x \in (a, b)$ there exists some $\xi \in (a, b)$ such that*

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

Proof. We first note that if $x = x_i$ for some $i = 0, \ldots, n$ then the equality holds. Thus we fix some $x \neq x_i$, and denote it as $x = x_{n+1}$. We consider the Newton form of the interpolant of f on $x_0, \ldots, x_{n+1}$, which is of the form

$$p_{n+1}(x) = p_n(x) + c_{n+1} \prod_{i=0}^{n} (x - x_i)$$

where p_n is the interpolant of f at $x_0, \ldots, x_n$. Therefore

$$f_{n+1}(x_{n+1}) = p_{n+1}(x_{n+1}) = p_n(x_{n+1}) + c_{n+1} \prod_{i=0}^{n} (x_{n+1} - x_i)$$

and thus

$$c_{n+1} = \frac{f(x_{n+1}) - p_n(x_{n+1})}{\prod_{i=0}^{n} (x_{n+1} - x_i)}.$$

Now note that $f - p_{n+1}$ is zero in $n+2$ distinct points $x_0, \ldots, x_{n+1}$ between a and b and therefore using Rolle's theorem we see that $f' - p'_{n+1}$ has $n+1$ distinct roots between a and b. Recursively applying this argument, we see that there exists some ξ between a and b such that

$$0 = f^{(n+1)}(\xi) - p_{n+1}^{(n+1)}(\xi) = f^{(n+1)}(\xi) - c_{n+1}(n+1)! = f^{(n+1)}(\xi) - (n+1)! \frac{f(x_{n+1}) - p_n(x_{n+1})}{\prod_{i=0}^{n} (x_{n+1} - x_i)}$$

By rearranging this expression we obtain what we wanted. □

Corollary 2. *Assume f is $n+1$ times continuously differentiable in an open neighborhood containing $[-1, 1]$, then for all $x \in [-1, 1]$]*

$$|f(x) - p_n(x)| = \frac{1}{(n+1)!} \max_{\xi \in [-1,1]} |f^{(n+1)}(\xi)| \max_{y \in [-1,1]} \prod_{i=0}^{n} |y - x_i|. \tag{27}$$

Remark: Note that For Taylor polynomials q_n focused around the point 0 we would get

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

and so a natural bound would be

$$|f(x) - q_n(x)| \leq \frac{1}{(n+1)!} \max_{\xi \in [-1,1]} |f^{(n+1)}(\xi)| \max_{y \in [-1,1]} |y|^{n+1}$$

Will this bound be better or worse than what we get from interpolation polynomials? The quality of the interpolation bound depends on the points x_i we choose. For example, we could get the Taylor bound exactly by choosing all x_i to be zero in (27). This is not really legal since the x_i must by different from each other, but one could instead choose them all to be in an interval $(-\epsilon, \epsilon)$ for some small ϵ and then we get that

$$\max_{y \in [-1,1]} \prod_{i=0}^{n} |(y - x_i)| \leq (1+\epsilon)^{n+1}$$

However we can also do better. What we will show is that there exists a choice of interpolation nodes x_i for which

$$\max_{y \in [-1,1]} \prod_{i=0}^{n} |y - x_i| = \left(\frac{1}{2}\right)^n$$

and this choice of x_i is the best possible choice. We get to this through Chebyshev polynomials:

Lesson 18

Remark: A monic polynomial p is a polynomial whose leading coefficient is one. A polynomial of the form

$$p(x) = \prod_{i=0}^{n} (x - x_i)$$

is monic. On the other hand, if p is degree $n+1$, is monic and has $n+1$ distinct real roots $x_i, i = 0, \ldots n$ in $[-1, 1]$ then

$$p(x) = \prod_{i=0}^{n} (x - x_i).$$

We will show that the Chebyshev polynomials are such polybomials, and their roots give us the best possible choice.

Chebyshev polynomials Chebyshev polynomials can be defined recursively via

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x), \forall n \geq 1 \end{aligned}$$

so we have

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 2x(2x^2 - 1) - x = 4x^3 - 3x \\ &\vdots \end{aligned}$$

Observations (i) T_n is a polynomial of degree n and (ii) the leading coeffecient in T_n is 2^{n-1}. Thus $\frac{1}{2^{n-1}} T_n$ is monic.

Recall that on the interval $[0, \pi]$ the consine function is injective and has an inverse (also known as arcos)

$$cos^{-1} : [-1, 1] \to [0, \pi]$$

we prove

Theorem 21. *For $x \in [-1, 1]$ and $n = 0, 1, 2, \ldots$ we have*

$$T_n(x) = \cos(n \cos^{-1}(x))$$

Proof. Denote

$$\tilde{T_n}(x) = \cos(n \cos^{-1}(x))$$

For $n = 0$

$$\tilde{T_0}(x) = \cos(0) = 1$$

For $n = 1$

$$\tilde{T_1}(x) = \cos(\cos^{-1}(x)) = x$$

Using the trigonometric identity

$$\cos((n+1)\theta) = 2\cos(\theta)\cos(n\theta) - \cos((n-1)\theta)$$

with $\theta = \cos^{-1} x$ we get

$$\tilde{T}_{n+1} = \cos((n+1)\cos^{-1}(x) = 2x\tilde{T_n} - \tilde{T}_{n-1}$$

so we see recursively that $T_n = \tilde{T_n}$. □

Lemma 11. *1. For every $x \in [-1, 1]$ we have $|T_n(x)| \leq 1$.*

2. For every $0 \leq j \leq n$

$$T_n(\cos(\frac{j\pi}{n})) = (-1)^j$$

3. The polynomial T_n has n roots in $[-1, 1]$: for every $1 \leq j \leq n$

$$T_n(\cos(\frac{(2j-1)\pi}{2n})) = 0$$

4. The leading coefficient of T_n is 2^{n-1}

Proof. The first three arguments follow easily from

$$T_n(x) = \cos(n \cos^{-1}(x))$$

and the last argument follows from the recursive formula as discussed earlier. □

We now have what we need to prove:

Theorem 22. *1. Let $\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_n$ be the roots of the Chebychev polynomial T_{n+1}. Then*

$$\max_{x \in [-1,1]} \prod_{i=0}^{n} |x - \bar{x}_i| = 2^{-n}$$

2. For every $n + 1$ distinct points $x_0, x_1, \ldots, x_n$ in $[-1, 1]$,

$$\max_{x \in [-1,1]} \prod_{i=0}^{n} |x - x_i| \geq 2^{-n}$$

Proof. 1. The leading coefficient of T_{n+1} is 2^n and so $2^{-n}T_{n+1}$ is monic and is the form

$$2^{-n}T_{n+1}(x) = \prod_{i=0}^{n}(x - \bar{x}_i)$$

and so for all $x \in [-1, 1]$

$$2^{-n} \geq 2^{-n}|T_{n+1}(x)| = \prod_{i=0}^{n} |x - \bar{x}_i|$$

and as there also exist x for which $|T_{n+1}(x)| = 1$ we have

$$\max_{x \in [-1,1]} \prod_{i=0}^{n} |x - \bar{x}_i| = 2^{-n}$$

2. For every $n+1$ distinct points $x_0, x_1, \ldots, x_n$ in $[-1,1]$ the polynomial

$$p(x) = \prod_{i=0}^{n} |x - x_i|$$

is monic of degree $n+1$. Assume by way of contradiction that

$$\max_{x\in[-1,1]} |p(x)| < 2^{-n}$$

then the polynomial $2^{-n}T_{n+1} - p$ is of degree n since $2^{-n}T_{n+1}$ and p have the same leading coefficient. Moreover we saw that they were $n+2$ points $x_j, j = 0, \ldots, n+1$ in $[-1,1]$ for which $T_{n+1}(x_j) = (-1)^j$ and since by assumption $|p(x_j)| < 2^{-n}$ we see that

$$sign((2^{-n}T_{n+1}(x_j) - p(x_j)) = (-1)^j$$

which implies by the intermediate value theorem that $2^{-n}T_{n+1} - p$ is a degree n polynomial with $n+1$ roots and so $p = 2^{-n}T_{n+1}$ and

$$\max_{x\in[-1,1]} |p(x)| = 2^{-n} \max_{x\in[-1,1]} |T_{n+1}(x)| = 2^{-n}$$

leading to a contradiction.

□

Conclusion Approximation a $n+1$ times continuosly differentiable function f on $[-1,1]$ by the Taylor polynomial of degree n developed around the point 0 has error

$$\max_{x\in[-1,1]} |f(x) - p_n(x)| \leq \frac{1}{(n+1)!} \max_{\xi\in[-1,1]} |f^{(n+1)}(\xi)| \max_{x\in[-1,1]} |x|^{n+1} = \frac{1}{(n+1)!} \max_{\xi\in[-1,1]} |f^{(n+1)}(\xi)|$$

We saw that if we approximate f by the degree n interpolant at the points $\bar{x}_0, \bar{x}_1, \ldots \bar{x}_n$ then the error we get is

$$\max_{x\in[-1,1]} |f(x) - p_n(x)| \leq \frac{1}{(n+1)!} \max_{\xi\in[-1,1]} |f^{(n+1)}(\xi)| \max_{x\in[-1,1]} \prod_{i=0}^{n} |x - x_i| = \frac{1}{2^n(n+1)!} \max_{\xi\in[-1,1]} |f^{(n+1)}(\xi)| \tag{28}$$

For function like $\sin(x), \cos(x), e^x$ for which

$$M_n = \max_{\xi\in[-1,1]} |f^{(n+1)}(\xi)|$$

does not grow with n this gives us an error bound which goes to zero very fast. But what happens for functions where M_n grows with n? Here is an example where that happens:

Example 1 Consider the function ($a > 0$)

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

we want to show that its $n+1$-th derivatives grow faster that the other terms in the bound in (28)

Proposition 14. *If $a > 4$ then the bound in (28) for the error of approximating $f(x) = \frac{1}{1+ax^2}$ on $[-1,1]$ with its interpolant of degree n at the Chebychev nodes, does not go to zero as $n \to \infty$.*

Proof. Recall that

$$\frac{1}{1-y} = \sum_{k=0}^{\infty} y^k, \forall y \in (-1,1)$$

and so

$$f(x) = \frac{1}{1-(-ax^2)} = \sum_{k=0}^{\infty} (-ax^2)^k = \sum_{k=0}^{\infty} (-a)^k x^{2k}, \quad \forall x \in (-\frac{1}{\sqrt{a}}, \frac{1}{\sqrt{a}})$$

In general, the $n+1$-th derivative of $\sum_k b_k x^k$ at zero is $b_{n+1}(n+1)!$, and so in our case we get that if $n+1$ is even,

$$f^{(n+1)}(0) = (-a)^{\frac{n+1}{2}}(n+1)!$$

plugging this into our error term we get

$$\frac{1}{2^n(n+1)!}\max_{\xi\in[-1,1]}|f^{(n+1)}(\xi)| \geq \frac{1}{2^n(n+1)!}|f^{(n+1)}(0)| = a^{\frac{n+1}{2}}2^{-n}$$
$$= a\left(\frac{\sqrt{a}}{2}\right)^n \to \infty \text{ as } n\to\infty \text{ if } a>4$$

□

We say that for smooth functions whose derivatives grow fast, the bounds we found may not be very useful. This is also the case for continuous functions which are not differentiable. How well can such functions be approximated by interpolation polynomials?

Recall that a sequence of continuous functions p_n converges uniformly in $[a,b]$ to a continuous function f if

$$\max_{x\in[a,b]}|f(x)-p_n(x)|\to 0$$

We have Wierstrass's theorem

Theorem 23 (Weirestrass, without proof)**.** *Assume f is continuous on $[a,b]$, then there exist a sequence of polynomials p_n which converges to f uniformly on $[a,b]$.*

So any continuous function can be approximated by polynomials. This does not necessarily mean that the polynomials obtained by interpolation will converge. Indeed Faber's theorem shows that this is not the case

Theorem 24 (Faber's theorem, without proof)**.** *For any prescribed system of nodes*

$$a\leq x_0^{(n)}<x_1^{(n)}<\ldots<x_n^{(n)}\leq b \tag{29}$$

there exists a continuous function f on $[a,b]$ such that the interpolation polynoials p_n do not converge to f uniformly.

Conclusions To conclude, we saw that interpolation polynomials (with Chebychev nodes) can give better bounds for approximating smooth functions, than would could obtain using Taylor polynomials. However if the derivatives grow very fast, or do not exists, interpolation may not give a good approximation. For this reason polynomial interpolation is not in widespread use in the present. A popular alternative is spline interpolation, where the function is interpolated on small intervals by a low degree polynomial. The simplest example is piecewise linear interpolation:

Lesson 18

Recall that if p is the degree n interpolation of (a sufficiently smooth) f at $n+1$ different points $x_0, \ldots, x_n$ in the interval $[a,b]$, then

$$|f(x) - p_n(x)| = \frac{1}{(n+1)!} \max_{\xi \in [a,b]} |f^{(n+1)}(\xi)| \max_{y \in [a,b]} \prod_{i=0}^{n} |y - x_i|, \forall x \in [a,b]. \tag{30}$$

In the last lesson we saw that this error bound may not go to zero when n increases, if the derivatives of f are not well controlled.

Piecewise Linear Interpolation Given $f : [a,b] \to \mathbb{R}$, the piecewise interpolant of f at $a = x_0 < x_1 < \ldots < x_n = b$ is the unique function S such that $S(x_i) = f(x_i), i = 0, \ldots, n$, and such that for all $i = 0, \ldots, n-1$ there exist $c_i, d_i \in \mathbb{R}$ such that

$$S(x) = c_i x + d_i, \forall x \in [x_i, x_{i+1}].$$

We prove that in this scheme, as the number of samples increases, we can always obtain convergence

Proposition 15. *Let $f : [a,b] \to \mathbb{R}$ be twice continuously differentiable in an open interval containing $[a,b]$, and let S be the piecewise linear interpolant of f at $a = x_0 < x_1 < \ldots < x_n = b$. Denote*

$$\delta = \max_i x_{i+1} - x_i, \text{ and } M = \max_{\eta \in [a,b]} |f''(\eta)|,$$

then

$$|f(x) - S(x)| \leq \frac{M}{8} \delta^2$$

This theorem implies that we can take the approximation error to zero by interpolating at more and more equispaced points. It only requires that the function is twice continuously differentiable.

Proof. Pick some arbitrary $x \in [a,b]$. There exists some i such that $x \in [x_i, x_{i+1}]$. In this interval the function S is the degree ≤ 1 interpolant of f at the points x_i, x_{i+1}. By Theorem 20

$$\begin{aligned}
|f(x) - S(x)| &\leq \frac{1}{2!} \max_{\eta \in [x_i, x_{i+1}]} |f''(\eta)| \max_{x \in [x_i, x_{i+1}]} |x - x_i||x - x_{i+1}| \\
&\leq \frac{M}{2} \max_{x \in [x_i, x_{i+1}]} |x - x_i||x - x_{i+1}| \\
&= \frac{M}{2} \left(\frac{x_{i+1} - x_i}{2} \right)^2 \\
&\leq \frac{M}{2} \frac{\delta^2}{4} = \frac{M}{8} \delta^2
\end{aligned}$$

□

Piecewise linear interpolants are called degree 1 splines. They are continuous but not differentiable. Often smoother splines are used: in general degree k splines will be $k-1$ times differentiable. Cubic splines ($k = 3$) which are twice differentiable are a popular choice. Before we define them let us discuss Hermite interpolation problems, which are of interest of their own right, but we can also think of them as a tool for showing that general splines are well defined.

4.3 Hermite interpolation

When we discussed polynomial interpolation, we saw that the interpolation equations

$$p(x_i) = y_i, i = 0, \ldots, n$$

are linear in the coeffecients of p. If we consider degree n polynomials we get $n+1$ linear equations in $n+1$ coordinates (the coefficients of p), and we proved that this linear equation has a unique solution.

We now consider polynomial interpolation problems which also involve derivatives, that is

$$p^{(k)}(x_0) = y_0$$

where x_0, y_0 are known. If we write

$$p(x) = c_0 + c_1 x + \ldots + c_n x^n$$

then this equation is linear in the coeffecients c_i.For example

$$p'(x_0) = y_0 \text{ if and only if } c_1 + 2c_2 x_0 + 3c_3 x_0^2 + \ldots + n c_n x_0^{n-1} = y_0$$

and more generally

$$p^{(k)}(x_0) = \sum_{j \geq k} \frac{j!}{(j-k)!} c_j x_0^{j-k}$$

Thus we could try to determine a degree $\leq n$ polynomial by setting $n+1$ values of the polynomial of its derivatives at a collection of points. Thus for example we can search for a polynomial $p \in P_2$ satisfying the interpolation conditions

$$p(0) = 0, p(1) = 1, p'(1/2) = 2$$

writing this in the standard basis $\{1, x, x^2\}$ of P_2 we get for

$$p(x) = c_0 + c_1 x + c_2 x^2$$

the equations

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$$

We see that the matrix we get is singular and the equation does not have a solution...

A large class of interpolation problems which do have a solution are Hermite interpolation problems:

Hermite interpolation problems: We are given distinct points

$$x_0, \ldots, x_n$$

and for each point x_i we are given k_i values

$$c_{ij}, \text{ where } j = 0, \ldots, k_i - 1$$

the goal is to find a polynomial p which for every x_i satisfies the k_i equations

$$p^{(j)}(x_i) = c_{ij}, 0 \leq j \leq k_i - 1$$

and has degree $\leq m$ where

$$m + 1 = k_0 + \ldots + k_n$$

examples The interpolation problem

$$p(0) = 0, p(1) = 1, p'(1/2) = 2$$

is not Hermite since $p(1/2)$ is not specified. In contrast

$$p(0) = 0, p(1/2) = 2, p'(1/2) = 2$$

is Hermite.

Theorem 25. *Every Hermite interpolation problem has a unique solution.*

Proof. Hermite interpolation problems involve $m+1$ linear equations in $m+1$ variables. It is sufficient to show that the homogenous equations have a unique solution, that is that the zero polynomial is the only $p \in \mathbb{R}_m[x]$ such that for every $x_i, i = 0, \ldots, n$,

$$p^{(j)}(x_i) = 0, 0 \leq j \leq k_i - 1$$

We first note that any polynomial p which satisfies these condition in x_i must divide $(x-x_i)^{k_i}$. To see this, we can divide p by $(x-x_i)^k$ with remainder to obtain

$$p(x) = (x-x_i)^k q(x) + r(x)$$

Assume by way of contradiction that r is not zero. Denote $deg(r) = d < k$. We then have that $p^{(d)}(x_i) = r^{(d)}(x_i) = 0$ which is a contradiction since the d-th derivative of a degree d polynomial is a non-zero constant.

Now, if p satisfied these conditions for all i it must be of the form

$$p(x) = q(x)\prod_{i=0}^{n}(x-x_i)^{k_i}$$

and since p is of degree $\leq m$ we must have $q = 0$ since otherwise the degree of p would be at least

$$k_0 + k_1 + \ldots k_n = m + 1.$$

□

4.4 Splines

A spline function S of degree k having knots $t_0 < t_1 < \ldots < t_n$ is a function satisfying

1. On each interval $[t_{i-1}, t_i), i = 1, \ldots, n$ the function S is a polynomial p_i of degree $\leq k$.
2. S has a continuous $k-1$ derivative on $[t_0, t_n]$.

Degree $k = 0$ splines are piecewise constant (typically discontinuous functions).

Degree $k = 1$ splines are continuous piecewise linear functions.

Our interest is in degree $k = 3$ splines which is the most commonly used. We call such splines cubic splines. We now consider interpolation problems where we specify arbitrarily values $(y_0, \ldots, y_n)$ to $(t_0, \ldots, t_n)$. As we will see even after adding these constraints we remain with two degrees of freedom which we can fix to get a unique cubicc spline. We claim as follows

Proposition 16. *For given distinct knots $(t_0, \ldots, t_n)$ and values $(y_0, \ldots, y_n)$ and values a_0, b_0, there exists a unique cubic spline s interpolating (t_j, y_j) and satisfying*

$$s'(t_0) = a_0$$
$$s''(t_0) = b_0$$

Proof. A cubic spline s is equal to a cubic polynomial p_i on each interval $[t_{i-1}, t_i)$. Therefore it can be identified with n-tuples $(p_1, p_2, \ldots, p_n) \in \mathbb{R}_3^n[x]$ which satisfy

$$p_j(t_j) = y_j j = 1, \ldots, n$$

$$p_j(t_{j-1}) = y_{j-1}, j = 1, \ldots, n$$
$$p_j'(t_j) = p_{j+1}'(t_j), j = 1, \ldots, n-1$$
$$p_j''(t_j) = p_{j+1}''(t_j), j = 1, \ldots, n-1$$

and additionally the initial conditions on the derivatives at t_0 must be satisfied. All in all the conditions on p_1 are then

$$p_1(t_0) = y_0$$
$$p_1(t_1) = y_1$$
$$p_1'(t_0) = a_0$$
$$p_1''(t_0) = b_0$$

this is a Hermite intepolation problem which determines p_1 uniquely. There are also conditions which involve p_1 and p_2. We can now proceed recursively- assuming we already know $p_1, \ldots, p_j$ for some $j \geq 1$, the conditions we see that p_{j+1} must satisfy

$$p_{j+1}(t_{j+1}) = y_{j+1}$$

$$p_{j+1}(t_j) = y_j$$
$$p'_{j+1}(t_j) = p'_j(t_j)$$
$$p''_{j+1}(t_j) = p''_j(t_j)$$

This is again a Hermite interpolation problem and there is a unique cubic p_{j+1} fulfilling these conditions. It follows that $(p_0, \ldots, p_n)$ are uniquely specified by these conditions and so there is a unique solution. □

Natural cubic splines We saw that two determine a cubic spline interpolant we need to specify two initial conditions. In this example there were the first and second derivative at t_0. There are other options. One common option are cubic splines which interpolate (t_j, y_j) and additionally satisfy the conditions

$$s''(t_0) = 0$$
$$s''(t_n) = 0.$$

These are called natural cubic splines.

Lesson 21

4.5 Least square approximation

Let us denote the space of continuous functions $f : [a, b] \to \mathbb{R}$ by $C([a, b], \mathbb{R})$, and the space of continuous complex valued functions $f : [a, b] \to \mathbb{C}$ by $C([a, b], \mathbb{C})$. When we want to say a statement which holds for both function spaces we will just write $C([a, b], F)$, where F is either $\mathbb{R}$ or $\mathbb{C}$.

The space of continuous functions $C([a, b], F)$ is a vector field over F, where addition of two function f, g is the function $f + g$ defined as

$$(f + g)(x) = f(x) + g(x)$$

and multiplication of a function f by a scalar $\lambda \in F$ is the function λf defined as

$$(\lambda f)(x) = \lambda f(x).$$

The zero in this vector space is the zero function $f(x) = 0, \forall x \in [a, b]$.

This vector space is infinite dimensional: for every n we have that the functions $1, x, x^2, \ldots, x^n$ are linearly independent, that is $\sum_k c_k x^k = 0$ if and only if $c_k = 0, \forall k$. Therefore the dimension is larger than n, for every finite n.

Our discussion so far was related to the uniform, or infinity, norm on $C([a, b], F)$, which is defined as

$$\|g\|_\infty = \sup_{x\in[a,b]} |g(x)|.$$

A norm is a measure of the length of an object in the vector space. The difference between two objects f, p in the vector space can be measured by the norm of their difference $\|f - p\|$. In our discussion so far we have attempted to get a good approximation of f by a polynomial p, in the sense that

$$\|f - p\|_\infty = \sup_{x\in[a,b]} |f(x) - p(x)|$$

is small. We've seen that interpolation can give us such an approximation when the derivatives of f exists and are well behaved, but in other cases his may fail. In general, the best thing we can do with degree $\leq n$ polynomials in terms of approximating f is to solve an optimization problem

$$\inf_{p\in\mathbb{R}_n[x]} \|f - p\|_\infty$$

We now from Weirstrass's theorem that the degree n polynomials p_n minimizing the expression above (assuming a minimum exists) would converge to zero. However, finding these best approximations is not easy to do. What we can do instead is try to find the best approximation with respect to other measure of distance on $C([a, b], F)$: norms which come from inner product spaces

The 2-norm on function spaces We can defined an inner product on $C([a, b], F)$ by

$$\langle f, g\rangle = \int_a^b f(x)\overline{g(x)}dx.$$

When $F = \mathbb{R}$ we just ignore the conjugation. When $F = \mathbb{C}$, the meaning of the integral is as follows: we write $h \in C([a, b], \mathbb{C})$ as $h(x) = h_1(x) + ih_2(x)$, where $h_1, h_2 \in C([a, b], \mathbb{R})$, and then

$$\int_a^b h(x)dx = \int_a^b h_1(x)dx + i\int_a^b h_2(x)dx$$

More generally, we can define for a continuous positive w on $[a, b]$ an inner product

$$\langle f, g\rangle = \int_a^b f(x)\overline{g(x)}w(x)dx$$

It is easy to see that this definition satisfies the inner product requirements

1. For all $f, h \in C([a, b], F)$ we have
$$\langle f, h\rangle = \overline{\langle h, f\rangle}$$

2. For all $f, h, g \in C([a,b], F)$ and $a, b \in \mathbb{R}$,

$$\langle ah + bg, f\rangle = a\langle h, f\rangle + b\langle g, f\rangle$$

3. For all $f \in C([a,b], F)$

$$\langle f, f\rangle \geq 0 \text{ with equality if and only if } f = 0$$

The inner product induces a norm

$$\|f\| = \sqrt{\langle f, f\rangle}.$$

When we use the standard function inner product we denote this norm by $\|\cdot\|_2$.

Given some $f \in C([a,b], F)$ and some notion of inner product on $C([a,b], \mathbb{C})$, we can search for the polynomial of degree $\leq n$ minimizing

$$\inf_{p \in \mathbb{R}_n[x]} \|f - p\|$$

as we will see, there always exists a minimizing polynomial and it can be found be solving a collection of linear equations.

Since we have different results with different norms, it is interesting to consider the relationship between the norms. We have the following simple lemma

Lemma 12. *If $f \in C([a,b], \mathcal{F})$ then*

$$\|f\|_2 \leq \sqrt{b-a}\|f\|_\infty$$

Proof. Since $\|f\|_\infty = \max_{x\in[a,b]} |f(x)|$ we have that

$$\|f\|_2^2 = \int_a^b |f(x)|^2 dx \leq \int_a^b \|f\|_\infty^2 dx = (b-a)\|f\|_\infty^2$$

□

Convergence On $C([a,b], F)$, we say that f_n converge to f with respect to some given norm $\|\cdot\|$ if $\|f - f_n\| \to 0$. When the norm we consider is ∞ we say that f_n converges to f uniformly, and when the norm is $\|\cdot\|_2$ we say that the convergence is in average.

Question 15. *1. Show that if $f_n \to f$ uniformly then $f_n \to f$ in average.*

2. Show that there exists polynomials p_n such that p_n converges to f uniformly.

3. Find f_n such that f_n converges to f in average, but not uniformly.

Finding best least square approximation We first remind ourselves of the Pythagorean theorem

Theorem 26 (Pythagoras)**.** *If C is an inner product space and $f, g \in C$ are orthogonal then*

$$\|f + g\|^2 = \|f\|^2 + \|g\|^2$$

Proof. Immediate from the fact that for general f, g (not necessarily orthogonal)

$$\begin{aligned}\|f + g\|^2 &= \langle f + g, f + g\rangle \\ &= \langle f, f + g\rangle + \langle g, f + g\rangle \\ &= \|f\|^2 + \|g\|^2 + 2\langle f, g\rangle\end{aligned}$$

□

Theorem 27. *Let C be an innner product space, $f \in C$, and P an n dimensional subspace with an orthogonal basis $\{u_1, \ldots, u_n\}$. Denote*

$$g = \sum_{i=1}^n \frac{\langle f, u_i\rangle}{\langle u_i, u_i\rangle} u_i.$$

Then

1. *$f - g$ is orthogonal to any element in P.*

2. *g is the **unique** best approximation of f in P. This is: for every $h \in P$ with $h \neq g$ we gave that $\|f - g\| < \|f - h\|$.*

Proof. For the first claim we have that for all $j = 1, \ldots, n$,

$$\langle u_j, f - g\rangle = \langle u_j, f\rangle - \frac{\langle u_j, f\rangle}{\langle u_j, u_j\rangle}\langle u_j, u_j\rangle = 0$$

and since this is true for all u_j it is true for their span which is P.

Next note that if $h \in P$ and $h \neq g$ then $f - g$ is orthogonal to all elements in P, and so also to $g - h$, and we obtain

$$\|f - h\|^2 = \|(f - g) + (g - h)\|^2 = \|f - g\|^2 + \|g - h\|^2 > \|f - g\|^2$$

□

Example Consider the inner product space $E = \mathbb{R}_2[x] \subseteq C([0, 1], \mathbb{R})$ with the inner product

$$\langle f, g\rangle = \int_0^1 f(x)g(x)dx$$

Define $f(x) = 10x + x^2$. We are searching for the closest linear polynomial in $\mathbb{R}_1[x]$ with respect to the inner product we defined. We will use an orthonormal basis $\{u_0, u_1\}$ obtained by applying Gram-Schmidt to the basis $1, x$. We will want to show that we again find the unique closest polynomial g we found before.

We denote $u_0 = 1$ which has $\|u_0\|^2 = \int_0^1 1 = 1$ and then take

$$u_1 = \frac{x - \langle x, 1\rangle 1}{\|x - \langle x, 1\rangle 1\|}$$

we see that $\langle x, 1\rangle = \int_0^1 xdx = \frac{1}{2}$ and

$$\|x - 1/2\|^2 = \|x\|^2 + 1/4\|1\|^2 - \langle x, 1\rangle = \int_0^1 x^2 + \frac{1}{4} - \int_0^1 xdx = 1/3 + 1/4 - 1/2 = \frac{1}{12}$$

so we get

$$u_0 = 1, u_1 = \sqrt{12}(x - \frac{1}{2})$$

the optimal linear polynomial which approximates of f is

$$g(x) = c_0 + c_1\sqrt{12}(x - \frac{1}{2})$$

where

$$c_0 = \langle f, u_0\rangle = \int_0^1 10x + x^2 dx = 5 + 1/3$$

and

$$\begin{aligned} c_1 = \langle f, u_1\rangle &= \sqrt{12}\int_0^1 10x^2 + x^3 - 5x - x^2/2dx = \sqrt{12}\,(10/3 + 1/4 - 5/2 - 1/6) \\ &= \sqrt{12}\frac{80 + 6 - 60 - 4}{24} = \frac{11\sqrt{12}}{12} = \frac{11}{\sqrt{12}} \end{aligned}$$

so

$$g(x) = 5\frac{1}{3} + 11(x - 1/2) = \frac{16}{3} - \frac{11}{2} + 11x = -\frac{1}{6} + 11x$$

We can verify that $f - g$ is orthogonal to every element in $\mathbb{R}_1[x]$ by checking (we will do this in class) that

$$\langle f - g, 1\rangle = 0$$

$$\langle f - g, x\rangle = 0$$

We'd also like to see that indeed $\|f-g\|$ is minimal. While it is difficult to verify this directly we can compare $\|f-g\|^2$ with $\|f-g_0\|^2$ where $g_0(x)=10x$ is a linear polynomial which seems a reasonable approximation for f. We see that

$$\|f-g_0\|^2 = \|x^2\|^2 = \frac{1}{5}$$

while

$$\|f-g\|^2 = \langle f-g, x^2-x+1/6\rangle = \langle f-g, x^2\rangle = \int x^4 - \int x^3 + 1/6 \int x^2$$
$$= \frac{1}{5} - \frac{1}{4} + \frac{1}{18} = \frac{36-45+10}{180} = \frac{1}{180}$$

Convergence

We now prove the following theorem:

Theorem 28. *Consider the standard inner product on $[a,b]$, choose some $f \in C[a,b]$ and let p_n be the best approximations of f in the subspace $P_n = \mathbb{R}_n[x]$. Then p_n converge to f w.r.t. the inner product norm.*

Proof. Since $\mathbb{R}_n[x] \subseteq \mathbb{R}_{n+1}[x]$ we know that $\|f-p_n\| \geq \|f-p_{n+1}\|$ so the sequence $\|f-p_n\|$ is monotonely non-decreasing and bounded from below by zero, and hence has some limit $L \geq 0$. We need to show that $L=0$. For every $\epsilon > 0$ by the Weierstrass theorem there exists a polynomial q s.t. the uniform error $\|f-q\|_\infty$ is smaller than ϵ. As we saw last lesson, it follows that

$$L \leq \|f-p_n\| \leq \epsilon\sqrt{b-a}$$

and since this is true for every $\epsilon \geq 0$ we have that $L=0$. □

4.6 Algorithm for creating orthogonal polynomials

The weighted inner products on $C([a,b],\mathbb{R})$

$$\langle f,g\rangle = \int_a^b f(x)g(x)w(x)dx$$

have the following property: For every continuous function h

$$\langle f\cdot h, g\rangle = \int_a^b f(x)h(x)g(x)w(x)dx = \langle f, g\cdot h\rangle$$

in other words, the linear operator A_h on the space of continuous functions defined by $A_h(f) = f\cdot h$ is symmetric. This property, specifically for the function $h(x)=x$ is useful for efficient construction of orthogonal polynomials. Denoting $A(f)=xf$ we note that $\mathbb{R}_n[x] = span\{1, A1, A^2 1, \ldots, A^n 1\}$. Thus $\mathbb{R}_n[x]$ is a Krylov subspace of $C([a,b],\mathbb{R})$. We've seen in the conjugate gradient method that this fact can be useful for efficient computations of orthogonal bases and we'll see this again here.

Lemma 13. *Let $p_0,\ldots,p_{n-1}$ be degree $0,1,\ldots,n-1$ polynomials which are orthogonal with respect to some weighted inner product on $C([a,b],\mathbb{R})$. Then $xp_{n-1}(x)$ is orthogonal to $p_0,\ldots,p_{n-3}$*

Proof. For fixed $k, k=0,\ldots,n-3$ we have that $xp_k(x)$ is a polynomial of degree $k+1 \leq n-2$. Since $p_0,\ldots,p_{n-2}$ are $n-1$ orthogonal polynomials in $\mathbb{R}_{n-2}[x]$ they are a bases of $\mathbb{R}_{n-2}[x]$ and therefore $xp_k(x) = \sum_{j=0}^{n-2} c_j p_j(x)$ for an appropriate choice of c_j. Thus

$$\langle p_k, xp_{n-1}(x)\rangle = \langle xp_k, p_{n-1}(x)\rangle = \sum_{j=0}^{n-2} c_j \langle p_j, p_{n-1}\rangle = 0$$

□

Based on the lemma we can use the following algorithm for creating orthogonal polynomials with respect to some weighted inner product:

$$\begin{aligned}
p_0(x) &= 1 \\
p_1(x) &= x - \frac{\langle x, p_0\rangle}{\langle p_0, p_0\rangle} p_0(x) \\
p_n(x) &= xp_{n-1} - \frac{\langle xp_{n-1}, p_{n-1}\rangle}{\langle p_{n-1}, p_{n-1}\rangle} p_{n-1}(x) - \frac{\langle xp_{n-2}, p_{n-2}\rangle}{\langle p_{n-2}, p_{n-2}\rangle} p_{n-2}(x)
\end{aligned}$$

Using the lemma, it is easy to verify recursively that the polynomials p_j have degree j and are orthonormal. They are also monic (leading coefficient is one).

Truncated Fourier Series as best approximations

Recall that the space $C([0, 2\pi], \mathbb{C})$ of continuous functions $f : [0, 2\pi] \to \mathbb{C}$ is a vector space over $\mathbb{C}$ with the inner product

$$\langle f, g\rangle = \frac{1}{2\pi} \int_0^{2\pi} f(x)\overline{g(x)}dx.$$

Note that the integral of $f(x) = f_1(x) + if_2(x)$ is defined as

$$\int_0^{2\pi} f(x)dx = \int_0^{2\pi} f_1(x) + i \int_0^{2\pi} f_2(x)$$

and let P_N be the subspace

$$P_N = span\{e^{inx}, n = -N, -N + 1, \ldots, N - 1, N\}$$

Elements in this subspace are called *trigonometric polynomials of degree* N. We wish to find the best approximation for a given $f \in C[0, 2\pi]$ in the space P_N. This is most natural to do when the function we are considering has a 2π period, e.g., a function f representing heart beats, rising and falling tides, etc.

We use the notation $E_n(x) = e^{inx}$.

To do so we begin by the following lemma which shows that the basis we defined is actually an orthonormal one:

Lemma 14. *The basis*

$$\{E_n(x), \quad n = -N, -N + 1, \ldots, N - 1, N\}$$

is orthonormal.

Proof. Note that for all non-zero integer n,

$$\int_0^{2\pi} E_n(x)dx = \int_0^{2\pi} \cos(nx)dx + i \int_0^{2\pi} \sin(nx)dx = \left[\frac{\sin(nx)}{n}\right]_0^{2\pi} - i \left[\frac{\cos(nx)}{n}\right]_0^{2\pi} = 0$$

While

$$\int_0^{2\pi} E_0(x)dx = 2\pi$$

Accordingly

$$\langle E_n(x), E_m(x)\rangle = \frac{1}{2\pi} \int_0^{2\pi} E_n\overline{E_m(x)}(x)dx = \frac{1}{2\pi} \int_0^{2\pi} E_{n-m}(x)dx = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m \end{cases}$$

□

Corollary 3. *Let $f : [0, 2\pi] \to \mathbb{C}$ be a continuous function. Then the closest function to f in P_N is*

$$f_N(x) = \sum_{n=-N}^{N} c_n E_n(x), \text{ where } c_n = \frac{1}{2\pi} \int f(x)\overline{E_n(x)}dx$$

Proof. Since $\{E_n\}_{|n|\leq N}$ is an orthonormal basis to P_N, the closest function is $f_N(x) = \sum_{n=-N}^{N} \langle g, E_n\rangle E_n$. □

Lesson 22

In Lemma 14 we saw that $E_{-N}, \ldots, E_N$ form an orthonormal basis for P_N w.r.t. the inner product

$$\langle f, g \rangle = \frac{1}{2\pi} \int_0^{2\pi} f(x)\overline{g(x)}dx.$$

As we know, there can be many orthonormal bases. Here is another example of an orthonormal basis:

Proposition 17. *The functions*

$$\begin{aligned} h_0(x) &= 1 \\ h_n(x) &= \sqrt{2}\cos(nx), \ldots n = 1, \ldots, N \\ g_n(x) &= \sqrt{2}\sin(nx), \ldots n = 1, \ldots, N \end{aligned}$$

are an orthonormal basis for $C([0, 2\pi], \mathbb{C})$.

Proof. We know that $E_n, n = -N, \ldots, N$ form a basis and hence $\dim(P_N) = 2N + 1$. Thus if we show that the $2N + 1$ functions we defined are orthogonal then it will immediately follow that they are also a basis. Now note that

$$\cos(nx) = \frac{1}{2}\left(E_n(x) + E_{-n}(x)\right), n = 0, \ldots, N$$

$$\sin(nx) = \frac{1}{2i}\left(E_n(x) - E_{-n}(x)\right), n = 1, \ldots, N$$

We easily see that for $n \neq m$, since $\langle E_n, E_m \rangle = 0$ then also

$$\langle \cos(nx), \sin(mx) \rangle = 0 = \langle \sin(nx), \sin(mx) \rangle = \langle \cos(nx), \cos(mx) \rangle = \langle 1, \cos(nx) \rangle = \langle 1, \sin(nx) \rangle.$$

It remains only to verify that

$$\begin{aligned} \langle sin(nx), \cos(nx) \rangle &= \frac{1}{4i}\langle E_n(x) - E_{-n}(x), E_n(x) + E_{-n}(x) \rangle \\ &= \frac{1}{4i}\left(\langle E_n(x), E_n(x) \rangle - \langle E_{-n}(x), E_{-n}(x) \rangle + \langle E_n(x), E_{-n}(x) \rangle - \langle E_{-n}(x), E_n(x) \rangle\right) \\ &= \frac{1}{4i}(1 - 1 + 0 - 0) = 0 \end{aligned}$$

Thus we have shown orthogonality. It remains to show that all elements of the orthogonal basis have norm 1. Indeed

$$\|h_0\| = \|E_0\| = 1$$

and

$$\begin{aligned} \|h_n\|^2 &= 2\langle \frac{1}{2}\left(E_n(x) + E_{-n}(x)\right), \frac{1}{2}\left(E_n(x) + E_{-n}(x)\right) \rangle \\ &= \frac{1}{2}\left(\|E_n(x)\|^2 + \|E_{-n}(x)\|^2\right) = 1 \end{aligned}$$

A similar computation shows that

$$\|g_n(x)\| = 1$$

□

Changes of variables: Scaling The functions $E_n(x)$ are periodic functions on $[0, 2\pi]$, and we use them to approximate functions with respect to the inner product

$$\langle f, g \rangle_{[0,2\pi]} = \frac{1}{2\pi} \int_0^{2\pi} f(x)\overline{g(x)}dx.$$

Similarly, if we consider for some $L > 0$ the inner product

$$\langle f, g \rangle_{[0,L]} = \frac{1}{L} \int_0^{L} f(x)\overline{g(x)}dx,$$

then the functions $\tilde{E}_n = \exp(\frac{2\pi nxi}{L})$ will be a L-periodic orthonormal basis since

$$\langle \tilde{E}_n, \tilde{E}_m \rangle_{[0,L]} = \frac{1}{L}\int_0^L \exp(\frac{2\pi nxi}{L})\exp(\frac{-2\pi mxi}{L})dx = \frac{1}{2\pi}\int_0^{2\pi} \exp(nyi)\exp(-myi)dy = \langle E_n, E_m \rangle_{[0,2\pi]}$$

where we used the change of variables $y = \frac{2\pi}{L}x, dy = \frac{2\pi}{L}dx$.

Changes of variables: translation Recall that if f is an L periodic function $f(x) = f(x+L), \forall x \in \mathbb{R}$, then for all $a \in \mathbb{R}$ we have

$$\int_0^L f(x)dx = \int_a^{L+a} f(x)dx$$

because

$$\int_0^L f(x)dx = \int_0^a f(x)dx + \int_a^L f(x)dx = \int_L^{L+a} f(x)dx + \int_a^L f(x)dx = \int_a^{L+a} f(x)dx$$

It follows that if $\tilde{E}_n$ is some orthonormal basis of L periodic functions with respect to the inner product $\langle f, g \rangle_{[0,L]}$, then it is also an orthonormal basis with respect to the inner product

$$\langle f, g \rangle_{[a,L+a]} = \frac{1}{L}\int_a^{L+a} f(x)\overline{g(x)}dx,$$

Question 16. *Find the best approximation for $f(x) = x$ in the subspace P_N in terms of the standard inner product on $[-\pi, \pi]$*

Answer 12. *We saw that*

$$\{1, \sqrt{2}\cos(x), \ldots, \sqrt{2}\cos(Nx), \sqrt{2}\sin(x), \ldots, \sqrt{2}\sin(Nx)\}$$

is an orthonormal basis for this inner product space and hence the best approximation is

$$g(x) = \langle f, 1\rangle 1 + \sum_{n=1}^{N}\langle f, \sqrt{2}\cos(nx)\rangle\sqrt{2}\cos(nx) + \sum_{n=1}^{N}\langle f, \sqrt{2}\sin(nx)\rangle\sqrt{2}\sin(nx)$$

Recall that if F is an odd function $F(-x) = -F(x)$ then $\int_{-\pi}^{\pi} F(x)dx = 0$. Using this we can easily see that

$$g(x) = \sum_{n=1}^{N} 2\langle f, \sin(nx)\rangle \sin(nx)$$

Now note that, using the change of variables integration formula

$$\int uv' = |uv - \int u'v \text{ with } u(x) = x, u'(x) = 1, v(x) = -\frac{\cos(nx)}{n}, v'(x) = \sin(nx)$$

we obtain

$$\langle f, \sin(nx)\rangle = \frac{1}{2\pi}\int_{-\pi}^{\pi} x\sin(nx)dx = \frac{1}{2\pi}\left(-\left[\frac{x\cos(nx)}{n}\right]_{-\pi}^{\pi} + \frac{1}{n}\int_{-\pi}^{\pi}\cos(nx)dx\right) = -\frac{\cos(n\pi)}{n} = \frac{(-1)^{n+1}}{n},$$

where we used the fact that $\int_{-\pi}^{\pi}\cos(nx) = 2\pi\langle 1, \cos(nx)\rangle = 0$. The best approximation for f is

$$g(x) = 2\sum_{n=1}^{N}\frac{(-1)^{n+1}}{n}\sin(nx)$$

Question 17. *Find the best approximation for $f_2(x) = x^2$ in the subspace P_N in terms of the standard inner product on $[-\pi, \pi]$*

Answer 13. *We do not compute the full answer. We do note that since $x^2\sin(nx)$ is odd the coefficients of $\sin(nx)$ in the expansion vanish. The best approximation is*

$$g_2(x) = \frac{\pi^2}{3} + 4\sum_{n=1}^{N}\frac{(-1)^n}{n^2}\cos(nx)$$

Do the approximations for x and x^2 converge as $N \to \infty$? We will look at this visually in class. As we will next see, for x we have convergence in average, but not uniform convergence. For x^2 we will have both kinds of convergence. The essential difference between them is that x^2 assumes the same value at the end points $-\pi$ and π.

Convergence and Fourier series We defined for $f \in C([0,2\pi],\mathbb{C})$ and $n \in \mathbb{Z}$ the Fourier coeffecients

$$c_n = \langle f, E_n(x) \rangle$$

and we saw that for a given $N \in \mathbb{N}$, the best approximation for f in P_N with respect to the standard inner product is

$$g_N = \sum_{|n| \leq N} c_n E_n$$

Does g_N converge to f? And if so in what sense? This much we can say easily

Proposition 18. *For $f \in C([0,2\pi],\mathbb{C})$, assume that there exists some $q_N \in P_N$ such that q_N converges to f on average. Then g_N converges to f on average.*

Proof. Since g_N is the best approximation in P_N we have that

$$\|f - g_N\|_2 \leq \|f - q_N\|_2 \to 0.$$

□

We leave the following theorem to a course in functional analysis

Theorem 29 (without proof)**.** *For every $f \in C([0,2\pi],\mathbb{C})$, the best approximation for f in P_N, which we denote by g_N, converges to f in average.*

Due to this theorem, it is customary to write that

$$f = \sum_{n \in \mathbb{Z}} c_n E_n := \lim_{N \to \infty} g_N = \sum_{|n| \leq N} c_n E_n.$$

The expression on the right hand sides is called the Fourier Series of f. Note that the convergence here is in the L^2 norm. Thus for example, using the Fourier coeffecients we computed earlier we can write for $f(x) = x$ that

$$f(x) = x = 2 \sum_{n=1}^{N} \frac{(-1)^{n+1}}{n} \sin(nx) \text{ as functions}$$

but note that this only means that the finite sums converge to f on average. We do not necessarily have uniform convergene, or even pointwise convergence of the series on the right to the expression on the left. Indeed this cannot be since $f(\pi) \neq f(-\pi)$. When this obstruction is lifted, we have uniform convergence under rather general conditions

Theorem 30 (without proof)**.** *If $f : [0,2\pi] \to \mathbb{C}$ is continuous, $f(0) = f(2\pi)$, and f' exists and is piecewise continuous in $[0,2\pi]$, then g_N converges to f uniformly.*

Application: Heat Equation In 1822, Jean-Baptiste Joseph Fourier suggested Fourier series as a means of solving the heat equation with periodic boundary conditions. In essence, this uses two useful properties of the trigonometric functions: The first is that they are eigenfunctions of the Laplacian, that is

$$\frac{d^2}{dx^2}\sin(nx) = -n^2\sin(nx) \text{ and } \frac{d^2}{dx^2}\cos(nx) = -n^2\cos(nx),$$

the second is periodicity. In more detail, assume we have a 'one-dimensional' metal rod of length 2π. Let $u(x,t) : [-\pi,\pi]\times[0,\infty) \to \mathbb{R}$ denote the heat of the rod at position x at time t, and assume that the endpoint of the rod are both held in the same condition so

$$u(-\pi,t) = u(\pi,t) \text{ and } u_x(-\pi,t) = u_x(\pi,t) \tag{31}$$

We also assume that the initial heat at time $t = 0$ is given, denoted by some known $f(x)$, that is

$$u(x,t) = f(x). \tag{32}$$

The propagation of the heat is described by the heat equation

$$u_t = ku_{xx} \tag{33}$$

Let us first find solutions to equations (33) and (31). We note that for every n, the constant function $u_0(x) = 1$ and the functions

$$u_n^{(1)}(t) = \exp(-n^2kt)\cos(nx)) \text{ and } u_n^{(2)}(t) = \exp(-n^2kt)\sin(nx))$$

solve these two equations. Moreover any linear combination of these equations $a_0u_0+\sum_{n=1}^{N} a_n\cos(nx))\exp(-n^2kt)+ b_n\sin(nx))\exp(-n^2kt)$ will also be a solution. For the last equation (32), We note that if f is a finite trigonometric polynomial

$$f(x) = a_0u_0 + \sum_{n=1}^{N} a_n\cos(nx)) + b_n\sin(nx)),$$

so for that type of initial condition, we have a solution for the heat equation. Fourier's argument was that any periodic function could be written as such an infinite sum

$$f(x) = a_0u_0 + \sum_{n=1}^{\infty} a_n\cos(nx)) + b_n\sin(nx))$$

and thus for any periodic function we found a solution. This argument requires some regularity assumptions on f to ensure convergence as in 30, and ensure that we can change the order of summation and differentiation.This type of questions are will not be addressed in this course.

4.7 Discrete Fourier Transform (DFT)

Often in applications we do not have an explicit formulat for the function but rather samples of the functions

$$y_k = f(\frac{2\pi k}{K}), k = 0, 1, \ldots, K-1$$

Assume we want to compute the Fourier series $f(x) = \sum_{n\in\mathbb{Z}} c_n e^{inx}$, where $c_n = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-inx}dx$. Since we only have discrete samples, we can approximate c_n using a Riemman sum

$$c_n = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-inx}dx \approx \frac{1}{2\pi}\sum_{k=0}^{K-1}\frac{2\pi}{K}f(\frac{2\pi k}{K})e^{-\frac{2\pi ikn}{K}} = \frac{1}{K}\sum_{k=0}^{K-1} y_k e^{-\frac{2\pi ikn}{K}}$$

Let us define this approximation of c_n by

$$c_n \approx \frac{1}{K}d_n \text{ where } d_n := \sum_{k=0}^{K-1} y_k e^{-\frac{2\pi ikn}{K}}. \tag{34}$$

We note that the sequence $(d_n)_{n\in\mathbb{Z}}$ is periodic in the sense that

$$d_{n+K} = \sum_{k=0}^{K-1} y_k e^{-\frac{2\pi ik(n+K)}{K}} = \sum_{k=0}^{K-1} y_k e^{-\frac{2\pi ikn}{K}} = d_n$$

where we used the fact that $e^{2\pi ib} = 1$ for all $b \in \mathbb{Z}$. Accordingly the seemingly infinite sequence d_n can be stored using only $d_0, \ldots, d_{K-1}$.

Approximation Assume that f is periodic and continuously differentiable in $[0, 2\pi]$, so we saw that the Fourier series of f converges to f uniformly,

$$f(x) = \sum_{n\in\mathbb{Z}} c_n E_n(x).$$

We'd like to approximate f only using finite samples of the sequence $(d_n)_{n\in\mathbb{Z}}$. How would you do it? Perhaps the most natural idea would be (lets assume $K = 2M + 1$)

$$f(x) \stackrel{?}{\approx} \frac{1}{K} \sum_{n\in\mathbb{Z}} d_n E_n(x).$$

However, since d_n is a periodic sequence we see that this doesn't make sense, this series will never converge. Intuitively this also make sense: using K samples we are not likely to get a good approximation of highly oscillatory function E_n with $n > K$. Alternatively, recall that the equality above implies that

$$f(x) \approx \sum_{-M}^{M} c_m E_m(x)$$

and thus we can try to approximate f using

$$f(x) \approx \frac{1}{K} \sum_{-M}^{M} d_m E_m(x)$$

This does indeed work:

Theorem 31 ([**?**])**.** *Let f be a periodic and continuously differentiable in $[0, 2\pi]$, and let $(d_n^M)_{n\in\mathbb{Z}}$ denote the sequence obtained by taking Riemman sums evaluated at $K = 2M + 1$ points as described above. Then the function*

$$f_M(x) = \sum_{m=-M}^{M} d_n^M E_n(x)$$

converges to f uniformly as $M \to \infty$.

The theorem in [**?**] (Corollary 3.5) has weaker requirements than those presented here.

Definition 9. *The discrete Fourier transform $\mathcal{F} : \mathbb{C}^K \to \mathbb{C}^K$ is the linear mapping*

$$(d_0, \ldots, d_{K-1}) = \mathcal{F}(y_0, \ldots, y_{K-1})$$

defined in (34)*.*

The DFT is unitary Let us denote

$$\omega_K := \exp(-2\pi i/K)$$

In this paragraph we will drop the subscript of ω_K and replace it with just ω. We note that

$$d_n = \sum_{k=0}^{K-1} y_k \omega^{nk}.$$

Accordingly, $\mathcal{F}$ is a linear mapping represented by the matrix

$$U = \frac{1}{\sqrt{K}} \left(\omega^{nk}\right)_{0\le n,k\le K-1}, \text{ in the sense that } \mathcal{F}(y) = \sqrt{K} U y$$

In homework you will prove

Lemma 15. *The matrix U is unitary.*

New Lesson

Recall that we defined a linear function $\mathcal{F} : \mathbb{C}^K \to \mathbb{C}^K$ which takes a vector $\vec{y} = (y_0, \ldots, y_{K-1})$ to a vector $\vec{d} = (d_0, \ldots, d_{K-1})$

$$d_n := \sum_{k=0}^{K-1} y_k e^{-\frac{2\pi i k n}{K}} = \sum_{k=0}^{K-1} y_k \omega^{nk}.$$

where

$$\omega = \omega_K = e^{-\frac{2\pi i}{K}}$$

In other words, denoting $U \in \mathbb{C}^{K\times K}$ as the matrix with

$$U_{nk} = \frac{1}{\sqrt{K}}(\omega^{nk})$$

we have that $\mathcal{F}y = \sqrt{K}Uy$. In homework you will prove that U is unitary. It follows that $\mathcal{F}$ is invertible and its inverse is represented by the matrix $\mathcal{F}^{-1} = \frac{1}{\sqrt{K}}U^*$. Since

$$U^* = \frac{1}{\sqrt{K}}\left(\bar{\omega}^{nk}\right)_{0\le n,k\le K-1}$$

we obtain that

$$y = \mathcal{F}^{-1}d$$

implying that

$$y_k = \frac{1}{K}\sum_{n=0}^{K-1} d_n \bar{\omega}^{nk} = \frac{1}{K}\sum_{n=0}^{K-1} d_n e^{\frac{2\pi i k n}{K}} = \frac{1}{K}\sum_{n=0}^{K-1} d_n E_n(\frac{2\pi k}{K}) \tag{35}$$

So the inverse DFT is very similar to the DFT itself.

In particular we have

Corollary 4. *Let $f : [0, 2\pi] \to \mathbb{C}$ be a function and denote*

$$y_k = f(\frac{2\pi k}{K}), k = 0, \ldots, K-1.$$

The vector $\vec{d} = \mathcal{F}(\vec{y})$ is the unique vector of coeffecients of a function in span$\{E_0, \ldots, E_{K-1}\}$ which interpolates f at the nodes $\frac{2\pi k}{K}$. That is, for all $k = 0, \ldots, K-1$,

$$f(\frac{2\pi k}{K}) = \frac{1}{K}\sum_{n=0}^{K-1} d_n E_n(\frac{2\pi k}{K})$$

Proof. For given $\vec{y}$ and $\vec{d} = \mathcal{F}\vec{y}$, we have that $\vec{y} = \mathcal{F}^{-1}(\vec{d})$. From (35) we see that indeed $\vec{d}$ is the vector of coefficients we are looking for. Moreover if there is some other vector $\vec{c}$ with the same properties then $\vec{y} = \mathcal{F}^{-1}(\vec{c})$, implying that

$$0 = \mathcal{F}^{-1}(\vec{c} - \vec{d}) \text{ if and only if } \vec{c} - \vec{d} = \mathcal{F}(0) = 0.$$

□

Remark: The sequence of K points $\vec{d} = \mathcal{F}(\vec{y})$ defined above can be extended to an infinite K-periodic sequence $(d_n)_{n\in\mathbb{Z}}$ by defining $d_n := d_{n \mod K}$. It will then be true that if $K = 2M + 1$ then $\frac{1}{K}\sum_{n=-M}^{M} d_n E_n$ is the unique function in span$\{E_{-M}, \ldots, E_M\}$ interpolating f at the sample points $\frac{2\pi k}{K}$. We've quoted a theorem that says that these approximations will converge uniformly to (periodic differentiable) f. The approximations in Corollary 4 will not generally do as well in approximating f outside the sample points.

4.8 Fast Fourier Transform (FFT)

We can rewrite the calculation in (34) as

$$d_n = \sum_{k=0}^{K1} y_k \omega_K^{nk}, \text{ where } \omega_K = \exp(-2\pi i/K),$$

It will be useful to note that

$$\omega_{2K}^{2j} = \exp(-\frac{2\pi i}{2K} 2j) = \exp(-\frac{2\pi i}{K} j) = \omega_K^j \tag{36}$$

Since the Discrete Fourier transform is a linear mapping $\mathcal{F} : \mathbb{C}^K \to \mathbb{C}^K$, it can in general be computed in $O(K^2)$ operations. The *Fast Fourier Transform (FFT)* is a method for computing the DFT in $K \log K$ operations. It is based on the following claim, which shows that if $K = 2S$, then the DFT of a singal of length $K = 2S$ can be computed by computing the DFT of two sub-signals of length S.

Proposition 19. *[FFT] Let $y = y_0, \ldots, y_{2S-1} \in \mathbb{C}^{2S}$, and denote*

$$y^{(0)} = [y_0, y_2, \ldots, y_{2(S-1)}] \in \mathbb{C}^S, y^{(1)} = [y_1, y_3, \ldots, y_{2S-1}] \in \mathbb{C}^S$$

and

$$d = \mathcal{F}(y), d^{(0)} = \mathcal{F}(y^{(0)}), d^{(1)} = \mathcal{F}(y^{(1)}),$$

and recall that $d^{(0)}, d^{(1)}$ can be extended to an infinite S-periodic sequence. Then

$$d_n = d_n^{(0)} + \omega_{2S}^n d_n^{(1)} \forall n = 0, \ldots, 2S-1$$

Proof.

$$\begin{aligned} d_n &= \sum_{k=0}^{2S-1} y_k \omega_{2S}^{nk} \\ &= \sum_{k=0}^{S-1} y_{2k} \omega_{2S}^{2nk} + \sum_{k=0}^{S-1} y_{2k+1} \omega_{2S}^{n(2k+1)} \\ &= \sum_{k=0}^{S-1} y_k^{(0)} \omega_S^{nk} + \omega_{2S}^n \sum_{k=0}^{S-1} y_k^{(1)} \omega_S^{nk} \\ &= d_n^{(0)} + \omega_{2S}^n d_n^{(1)} \end{aligned}$$

□

We deduce from this argument that if the DFT of a signal of length S can be computed in $T(S)$ operations, then the DFT of a signal of length $2S$ can be computed using

$$T(2S) \leq 2T(S) + C \cdot S \tag{37}$$

operations (which include multiplications, summation, and exponentiation).

Now assume that $K = 2^L$ for some $L \in \mathbb{N}$. This assumption is convenient and not that restrictive since usually one can choose the number of samples. Nonetheless we note that this assumption is convenient but is not essential, modifications can be made to reserve the efficiency of the algorithm also without this assumption.

In any case, let us assume $K = 2^L$. We can compute the DFT for a signal of length K using 2 DFTs of signals of length 2^{L-1}, each one of these DFT can be computed using the DFTs of 2 signals of length 2^{L-2} etc. Let us denote the complexity of computing the DFT of a signal of length 2^L using this method by $T(2^L)$. We have from (37) that

$$T(2^L) \leq 2T(2^{L-1}) + C2^L, \forall L \in \mathbb{N} \tag{38}$$

We then prove

Lemma 16. *If (38) holds for all $L \in \mathbb{N}$ then*

$$T(2^L) \leq 2^L T(1) + CL2^L, \forall L = 0, 1, \ldots$$

Proof. By induction. For $L = 0$ we obtain equality. Assume correctness for some L, then

$$\begin{aligned} T(2^{L+1}) &\leq 2T(2^L) + C2^{L+1} \\ &\leq 2(2^L T(1) + CL2^L) + C2^{L+1} \\ &= 2^{L+1}T(1) + C(L+1)2^{L+1} \end{aligned}$$

□

We conclude that when $K = 2^L$, then

$$T(K = 2^L) \leq KT(1) + CK\log_2(K) = O(K\log_2(K))$$

The Fourier Transform and Fourier dictionary We complement our discussion of Fourier analysis with a short mention of the Fourier Transform, and by summarizing the various different concepts in Fourier analysis and the relationship between them.

The Fourier transform (FT) of a function $f : \mathbb{R} \to \mathbb{C}$ with bounded integral

$$\int_{-\infty}^{\infty} |f(x)|dx$$

is the function $\hat{f} : \mathbb{R} \to \mathbb{C}$ defined as

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\exp(-2\pi i\xi x)dx$$

Under certain (non-trivial) conditions on f we have that

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi)\exp(2\pi i\xi x)d\xi$$

We can interpert this as follows: f can be written as a weighted integral(=infinite sum) of waves $\exp(2\pi i\xi x)$. The weight assigned for each wave is the correlation between the function and the wave $\hat{f}(\xi) = \langle f, \exp(2\pi i\xi x)\rangle$.

For the **Fourier series**, we have a similar picture: but here we consider $f : [0, 2\pi] \to \mathbb{C}$ which is periodic, and (using the notation $c_n = \hat{f}(n)$)

$$f(x) = \sum_{n\in\mathbb{Z}} \hat{f}(n)e^{inx}$$

so f can be written as a weighted countable sum of waves (only those that have period 2π), and the weights are again the inner product between the function and the wave

$$\hat{f}(n) = \langle f, e^{inx}\rangle.$$

In the **Discrete Fourier Transform (DFT)** we have (denoting $y_k = f(\frac{2\pi k}{K})$and $d_n = \hat{f}(n)$) that

$$y_k = \frac{1}{K}\sum_{n=0}^{K-1} d_n E_n(\frac{2\pi k}{K})$$

so the function f sampled at K points, can be represented at those points as a sum of K waves

$$f(\frac{2\pi k}{K}) = \frac{1}{K}\sum_{n=0}^{K-1} \hat{f}(n)E_n(\frac{2\pi k}{K})$$

where $\hat{f}(n) = \sum_{k=0}^{K-1} f(\frac{2\pi k}{K})E_n(-\frac{2\pi k}{K})$ is the correlatoin with E_n measured at K points.

Finally, the **Fast Fourier Transform (FFT)** is just an algorithm for computing the DFT.

Lesson 23

5 Numerical Integration

There are some functions for which we know how to numerically compute an integral, e.g.

$$\int_0^1 e^{ax}dx = [e^{ax}/a]_0^1 = \frac{1}{a}(e^a - 1)$$

however, in some cases anti-differentiation may be difficult. The most known example is

$$f(x) = e^{-x^2}$$

Another example could be functions for which we have no explicit formula, and we only how to sample. This is similar to the motivation we gave for using the DFT.

A natural method for approximating function integrals using samples is the following:

Definition 10 (Quadrature rules)**.** *A quadrature rule for approximating integrals of functions f on $[a,b]$ is a mapping $\Lambda : C[a,b] \to \mathbb{R}$ (from now on we denote $C[a,b] = C([a,b],\mathbb{R})$) of the form*

$$\Lambda(f) \mapsto \sum_{k=0}^{n} A_k f(x_k)$$

where $A_k \in \mathbb{R}$ and $x_k \in [a,b]$ for $k = 0,\ldots,n$.

Remark Note that, like standard integrals, a quadrature rule is a linear mapping: if $\lambda \in \mathbb{R}$ and $f, g \in C[a,b]$ then

$$\Lambda(\lambda f) = \sum_{k=0}^{n} A_k \lambda f(x_k) = \lambda \sum_{k=0}^{n} A_k f(x_k) = \lambda\Lambda(f)$$
$$\Lambda(f+g) = \sum_{k=0}^{n} A_k (f(x_k) + g(x_k)) = \sum_{k=0}^{n} A_k f(x_k) + \sum_{k=0}^{n} A_k g(x_k) = \Lambda(f) + \Lambda(g)$$

One simple example of a quadrature rule is Riemann sums, where we assume we have a partition

$$a = t_0 < t_1 < \ldots < t_{n+1} = b,$$

and we take $x_k \in [t_k, t_{k+1}]$ and $A_k = t_{k+1} - t_k$. Since we know Riemann sums can approximate integrals of f for any integrable f, we know that quadrature rules can approximate integrals when n is large enough.

Definition 11 (Exact Quadrature rules)**.** *We say a quadrature rule is* Exact *on a function space C if for all $f \in C$.*

$$\int_a^b f(x)dx = \sum_{k=0}^{n} A_k f(x_k)$$

Proposition 20. *Given points $x_0,\ldots,x_n$ in $[a,b]$ there exist unique $A_1,\ldots,A_k \in \mathbb{R}$ such that for every $p \in \mathbb{R}_n[x]$*

$$\int_a^b p(x)dx = \sum_{k=0}^{n} A_k p(x_k)$$

Proof. The proof uses the interpolation polynomials in Lagrange form: Let $\ell_0,\ldots,\ell_n$ be the degree n polynomials satisfying

$$\ell_k(x_j) = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases}.$$

If $p \in \mathbb{R}_n[x]$ then from the uniqueness of the interpolation polynomial

$$p(x) = \sum_{k=0}^{n} p(x_k)\ell_k(x)dx.$$

and so setting $A_k = \int_a^b \ell_k(x)dx$ we have

$$\int_a^b p(x)dx = \sum_{k=0}^n p(x_k) \int_a^b \ell_k(x)(x)dx = \sum_{k=0}^n A_k p(x_k)$$

To show that $A_0, \ldots, A_n$ are unique: assume that there exist some A_k such that

$$\int_a^b p(x)dx = \sum_{k=0}^n A_k p(x_k), \forall p \in \mathbb{R}_n[x]$$

then this in particular is true for $p = \ell_j$ for every $j = 0, \ldots, n$ and so

$$\int_a^b \ell_j(x)dx = A_j.$$

□

Low dimensional examples

Newton Cotes formula is of the form discussed in the previous proposition, where the points $x_0, \ldots, x_n$ are equally spaced. We consider two important special cases where $n = 1$ and $n = 2$.

Trapezoid rule (n=1). In the trapezoid rule we approximate integrals on $[a, b]$ by choosing $x_0 = a, x_1 = b$, and so we get quadrature rules of the form

$$A_0 f(a) + A_1 f(b),$$

which will be exact on $\mathbb{R}_1[x]$ for

$$A_0 = \int_a^b \ell_0(x)dx = \int_a^b \frac{b-x}{b-a} = \frac{1}{b-a}\left(b(b-a) - b^2/2 + a^2/2\right) = b - \frac{a+b}{2} = \frac{b-a}{2}$$

Similarly

$$A_1 = \int_a^b \ell_1(x)dx = \int_a^b \frac{x-a}{b-a}dx = \frac{b-a}{2}$$

so the trapezoid quadrature rule is

$$f \mapsto \frac{b-a}{2}[f(a) + f(b)]$$

We see that for $f(a), f(b) > 0$ the quadrature rule measures the area of the trapezoid formed by the graph of the linear interpolant of f at a, b. Indeed, this is not surprising since this is the integral of the interpolant and we saw the quadrature rule is exact on $\mathbb{R}_1[x]$.

Simpson's rule ($n = 2$) For given $a < b$ we wish to find the coefficients A_0, A_1, A_2 such that for all $p \in \mathbb{R}_2[x]$

$$\int_a^b p(x)dx = A_0 p(a) + A_1 p\left(\frac{a+b}{2}\right) + A_2 p(b)$$

let us first do this for $a = -1, b = 1$. Since the integral, and quadrature rules, are linear functions, it is sufficient to find A_0, A_1, A_2 for which this equation is satisfies for each basis element $\{1, x, x^2\}$ of $\mathbb{R}_2[x]$. This gives us equations

$$\begin{aligned} 2 &= \int_{-1}^1 1dx = A_0 + A_1 + A_2 \\ 0 &= \int_{-1}^1 xdx = -A_0 + A_2 \\ \frac{2}{3} &= \int_{-1}^1 x^2 dx = A_0 + A_2 \end{aligned}$$

the solution is

$$A_0 = A_2 = \frac{1}{3}, A_1 = \frac{4}{3}$$

so when $n = 2$ the quadrature rule we obtain for the interval $[-1, 1]$ at the points $-1, 0, 1$ is

$$f \mapsto \frac{1}{3}(f(-1) + 4f(0) + f(1))$$

by construction, this formula is exact on $\mathbb{R}_2[x]$. In fact, it is also exact on $\mathbb{R}_3[x]$. To see this we just need to see that the formula is exact for x^3. Indeed

$$0 = \int_{-1}^{1} x^3 dx = \frac{1}{3}(-1 + 0 + 1) = 0$$

We now derive the formula for general $[a, b]$: there is a linear monotonely increasing bijection $\lambda : [-1, 1] \to [a, b]$ given by

$$\lambda(x) = \frac{b-a}{2}x + \frac{b+a}{2}$$

using the change of variable formula with $y = \lambda(x), dy = \frac{b-a}{2}dx$ we get for every $f \in \mathbb{R}_3[x]$ that

$$\begin{aligned}\int_a^b f(y)dy &= \int_{-1}^{1} f(\lambda(x))\frac{b-a}{2}dx \\ &= \frac{b-a}{6}\left(f(\lambda(-1)) + 4f(\lambda(0)) + f(\lambda(1))\right) \\ &= \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)\end{aligned}$$

Composite Simpson Rule As before, we decompose the integral on $[a, b]$ into a sum of integral over smaller intervals and approximate each integral using Simpson's rule.

Gaussian quadrature

Recall that in Simpson's rule we were using a quadrature rule with points $x_0 = a, x_1 = \frac{a+b}{2}, x_2 = b$ to get exactness for $\mathbb{R}_3[x]$ and not only $\mathbb{R}_2[x]$. Can we reproduce this 'miracle' in general? It turns out we can actually do better: while we know that for every choice of $x_0, \ldots, x_n$ we can get exactness in $\mathbb{R}_n[x]$, there are special choices of $x_0, \ldots, x_n$ for which we can get exactness in $\mathbb{R}_{2n+1}[x]$. These special points are roots of orthogonal polynomials. We begin by proving the following theorem which shows that orthgonal polynomials always have $n + 1$ roots in the interval of integration:

Theorem 32. *If q be a degree $n + 1$ polynomial, orthogonal to $\mathbb{R}_n[x]$ in $[a, b]$, in the sense that*

$$\int_a^b q(x)p(x)dx = 0, \forall p \in \mathbb{R}_n[x]$$

then q has $n + 1$ roots in $[a, b]$.

Proof. Let $x_0, \ldots, x_N$ be the roots of q which are real and in the interval $[a, b]$. We know that $N \leq n$, we want to show that $N = n$. We know that there exist natural numbers $r_0, \ldots, r_N \geq 1$ and a polynomial Q with real coefficients such that

$$q(x) = Q(x)\prod_{i=0}^{N}(x - x_i)^{r_i}$$

Our goal now is to show that there exits a polynomial p of degree at most $N + 1$ so that $q(x)p(x) \geq 0$ for all $x \in \mathbb{R}$. To do this, note that since Q has no roots in $[a, b]$, then either $Q(x) > 0 \forall x \in [a, b]$ or $Q(x) < 0, \forall x \in [a, b]$. Thus we can choose a scalar $s \in \{-1, 1\}$ such that $sQ(x) > 0, \forall x \in [a, b]$. Next, for $i = 0, \ldots, N$ we define

$$t_i = \begin{cases} 1 & \text{if } r_i \text{ is odd} \\ 0 & \text{if } r_i \text{ is even} \end{cases}$$

so that $t_i + r_i$ is even for all i. We now have that

$$p(x) = s\prod_{i=0}^{N}(x - x_i)^{t_i}$$

is a polynomial of degree $\leq N+1$, and

$$p(x)q(x) = sQ(x) \cdot \prod_{i=0}^{N}(x-x_i)^{t_i+r_i} \geq 0, \forall x \in [a,b]$$

and since $p(x)q(x)$ is strictly positive for $x \neq x_i, i = 0, \ldots, N$ we have that

$$\int_a^b p(x)q(x) > 0$$

On the other hand, if $N < n$ then p has degree $N+1 \leq n$ and we'd get a contradiction to orthogonality $\int_a^b p(x)q(x) = 0$. Therefore $N = n$. □

Theorem 33. *Let q be a a polynomial of degree $n+1$ which is orthogonal to $\mathbb{R}_n[x]$ over $[a,b]$ in the sense that*

$$\int_a^b q(x)p(x)dx = 0, \forall p \in \mathbb{R}_n[x].$$

Let q be the $n+1$ distinct real roots $x_0, \ldots, x_n$ in $[a,b]$, then the unique quadrature formula defined by these points

$$f \mapsto \sum_{k=0}^{n} A_k f(x_k)$$

which is exact on $\mathbb{R}_n[x]$, is also exact on $\mathbb{R}_{2n+1}[x]$.

Proof. Let $f \in \mathbb{R}_{2n+1}[x]$. We can divide f by the degree $n+1$ polynomial q with remainder and get

$$f = qp + r$$

where $p, r \in \mathbb{R}_n[x]$. At the roots $x_0, \ldots, x_n$ of q we get $f(x_i) = r(x_i)$. The quadrature rule defined by the points $x_0, \ldots, x_n$ is *exact* on $\mathbb{R}_n[x]$. Therefore

$$\int_a^b f(x)dx = \int_a^b q(x)p(x)dx + \int_a^b r(x)dx = \int_a^b r(x)dx = \sum_{i=0}^{n} A_i r_i(x) = \sum_{i=0}^{n} A_i f_i(x)$$

□

Gaussian quadrature: is a quadrature rule as in the theorem:

1. we find a degree $n+1$ polyonial q which is orthogonal to $\mathbb{R}_n[x]$ with respect to the inner product $\int_a^b f(x)dx$
2. we find its roots $x_0, \ldots, x_n$ which are all in $[a,b]$
3. we compute $A_0, \ldots, A_n$ so that $\sum_{k=0}^{n} A_k f(x_k)$ is exact on $\mathbb{R}_n[x]$. By the thoerem this will also be exact on $\mathbb{R}_{2n+1}[x]$.

Convergence of Gaussian quadrature is proved using the following lemma:

Lemma 17. *Let $x_0, \ldots, x_n \in [a,b]$ be distinct points and $A_0, \ldots, A_n$ such that*

$$\int_a^b p(x) = \sum_{k=0}^{n} A_k p(x_k), \forall p \in \mathbb{R}_{2n+1}[x],$$

then the coeffiecients $A_k, k = 0, \ldots, n$ are non-negative and their sum is $b-a$.

Proof. The Gaussian quadrature formula is exact for $1 \in \mathbb{R}_{2n+1}[x]$ and so

$$b - a = \int_a^b 1dx = \sum_{k=0}^{n} A_k$$

Fix some $j, 0 \leq j \leq n$. Then

$$p(x) = \prod_{k \neq j} (x - x_k)^2$$

is a non-negative, non-zero polynomial of degree $2n$, and therefore by exactness

$$0 < \int_a^b p^2(x)dx = \sum_{k=0}^{n} A_k p^2(x_k) = A_j p^2(x_j)$$

so $A_j > 0$. □

Last lesson

We can now prove convergence

Theorem 34. *Let Let $f : [a, b] \to \mathbb{R}$ be a continuous functions and $P \in \mathbb{R}_{2n+1}[x]$ a polynomial such that*

$$|f(x) - P(x)| \leq \epsilon, \forall x \in [a, b].$$

Let $x_0, \ldots, x_n \in [a, b]$ be distinct points and $A_0, \ldots, A_n$ such that

$$\int_a^b p(x) = \sum_{k=0}^{n} A_k p(x_k), \forall p \in \mathbb{R}_{2n+1}[x].$$

Then

$$|\int_a^b f(x)dx - \sum_{k=0}^{n} A_k f(x_k)| \leq 2\epsilon(b - a)$$

Proof. Using the previous lemma we get

$$\begin{aligned}
\left|\int_a^b f(x)dx - \sum_{k=0}^{n} A_k f(x_k)\right| &\leq \left|\int_a^b f(x)dx - \sum_{k=0}^{n} A_k P(x_k)\right| + \left|\sum_{k=0}^{n} A_k \left(P(x_k) - f(x_k)\right)\right| \\
&= \left|\int_a^b (f(x) - P(x))dx\right| + \left|\sum_{k=0}^{n} A_k \left(P(x_k) - f(x_k)\right)\right| \\
&\overset{(*)}{\leq} \int_a^b |f(x) - P(x)|dx + \sum_{k=0}^{n} |A_k||P(x_k) - f(x_k)| \\
&\leq \epsilon \int_a^b 1dx + \epsilon \sum_{k=0}^{n} A_k = 2\epsilon(b - a)
\end{aligned}$$

□

Approximation rate We've seen that the integral of a continuous function can be approximated by Guassian quadratures with enough points. This is also true if we just use Riemann sums. Another important method is using 'composite integration rules'. The idea here is that we divide $[a, b]$ into subintervals of size h, and on each small interval $[a', b' = a' + h]$ we will apply a quadrature rule. To a certain extent the quality of such a rule is measured by the dependence of the error on h.

Recall from Theorem 20 that if $f \in C^{(N+1)}[a, b]$, the points $x_0, \ldots, x_N$ are distinct points in $[a, b]$ and p is the degree $\leq N$ interpolation polynomial, then for every $x \in [a, b]$ there exist $\xi \in (a, b)$ such that

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

as we've discussed, this can lead to very good approximation bounds for some functions, but when $f^{(N+1)}(\xi)$ grows fast the bounds may not go to zero as N grows. This can give us the following bound on the error

Lemma 18. *Let $n, N \in \mathbb{N}$ with $n \leq N$, and let $f \in C^{N+1}[a,b]$ and $x_0, \ldots, x_n$ be distinct points in $[a,b]$ and $A_0, \ldots, A_n \in \mathbb{R}$ such that*

$$\int_a^b p(x)dx = \sum_{k=0}^{n} A_k p(x_k), \forall p \in \mathbb{R}_N[x]$$

Then

$$|\int_a^b f(x) - \sum_{k=0}^{n} A_k f(x_k)| \leq \frac{(b-a)^{N+2}}{(N+1)!} \max_{\xi \in [a,b]} |f^{(N+1)}(\xi)|$$

Note this lemma applies to the general case where $N = n$ (which we can obtain for any choices of $x_0, \ldots, x_n$) and also for cases like Gaussian quadrature where $N = 2n + 1$.

Proof. Let $p \in \mathbb{R}_N[x]$ an interpolation polynomial of f at $N+1$ points $x_0, \ldots, x_N$ where the points $x_{n+1}, \ldots, x_N$ are chosen arbitrarily in $[a,b]$. We know from Theorem 20 that for every $x \in [a,b]$ there exist $\xi \in (a,b)$ such that

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

Since $|x - x_i| \leq b - a$ we get

$$|f(x) - p(x)| \leq \frac{(b-a)^{N+1}}{(N+1)!} \max_{\xi \in [a,b]} |f^{(N+1)}(\xi)|$$

and so

$$\begin{aligned}
|\int_a^b f(x) - \sum_{k=0}^{n} A_k f(x_k)| &= |\int_a^b f(x) - \sum_{k=0}^{n} A_k p(x_k)| \\
&= |\int_a^b (f(x) - p(x))dx| \leq \int_a^b |f(x) - p(x)|dx \\
&\leq \frac{(b-a)^{N+1}}{(N+1)!} \max_{\xi \in [a,b]} |f^{(N+1)}(\xi)| \int_a^b 1 dx \\
&= \frac{(b-a)^{N+2}}{(N+1)!} \max_{\xi \in [a,b]} |f^{(N+1)}(\xi)|
\end{aligned}$$

□

Let us consider the meaning of this bound. Note that $\frac{(b-a)^{N+2}}{(N+1)!} \to 0$, so that if the derivative of f do not grow very fast with N we will get convergence. As we saw in the past, there are examples where we do get such fast growth. This can be handled using composite integration error. **Composite integration error** Assume for some K we define $h = \frac{b-a}{J}$ and

$$a = t_0, t_1 = a + h, t_2 = a + 2h, \ldots, t_J = a + Kh = b$$

we write the integral of f as a sum of integrals on intervals of size h,

$$\int_a^b f(x)dx = \sum_{j=1}^{J} \int_{t_{j-1}}^{t_j} f(x)dx$$

and on each interval we approximated the integral by a quadrature rule $\sum_{k=0}^{n} A_k^j f(x_k^j)$ which is exact for polynomials

in $\mathbb{R}_N[x]$. Then we obtain the following estimate for the error

$$\begin{aligned}
|\int_a^b f(x)dx - \sum_{j=1}^{J}\sum_{k=0}^{n} A_k^j f(x_K^j)| &= |\sum_{j=1}^{J}\int_{t_{j-1}}^{t_j} f(x)dx - \sum_{k=0}^{n} A_k^j f(x_K^j)| \\
&\leq \sum_{j=1}^{J} |\int_{t_{j-1}}^{t_j} f(x)dx - \sum_{k=0}^{n} A_k^j f(x_k^j)| \\
&\leq \sum_{j=1}^{J} \frac{h^{N+2}}{(N+1)!} \max_{\xi\in[t_{j-1},t_j]} |f^{(N+1)}(\xi)| \\
&\leq \sum_{j=1}^{J} \frac{h^{N+2}}{(N+1)!} \max_{\xi\in[a,b]} |f^{(N+1)}(\xi)| \\
&= (b-a)\frac{h^{N+1}}{(N+1)!} \max_{\xi\in[a,b]} |f^{(N+1)}(\xi)|
\end{aligned}$$

What is the meaning of this: Once we fix N, we can take h to be smaller and smaller and get better and better approximations. This works even with $N = 0$ or $N = 1$, but we get better convergence order when N is larger.

Example: Composite trapezoid rule

Assume for some J we define $h = \frac{b-a}{J}$ and

$$a = t_0, t_1 = a + h, t_2 = a + 2h, \ldots, t_J = a + Jh = b$$

and we approximate

$$\int_a^b f(x)dx = \sum_{j=1}^{J}\int_{t_{j-1}}^{t_j} f(x)dx$$

by

$$\sum_{j=1}^{J} \frac{1}{2}(t_{j+1} - t_j)[f(t_j) + f(t_{j+1})] = \frac{h}{2}[f(t_0) + 2\sum_{j=1}^{J-1} f(t_j) + f(t_J)]$$

Since the trapezoid rule is exact for polynomials of degree $\leq N$ the approximation rate of this approximation rule will be proportional to $h^{N+1} = h^2$

References