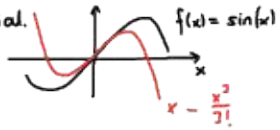
APPROXIMATION THEORY

key idea: Replace a complicated function by a simple function.

e.g. 1). $\pi \longrightarrow 3.14$.

2). function $f(x) \longrightarrow$ tangent $f'(x)$

3). function $f(x) \longrightarrow$ Taylor polynomial. $\sum_{k=0}^{n} \frac{f^{(k)}(0)}{n!} x^n$



4). All of science/engineering measurements $\longrightarrow$ finite precision.

Why are simple functions better?

- Easier to store (discrete representation – e.g. mp3 or jpeg compression).
- Easier to evaluate.
- Easier to compute with (e.g. differentiate, find roots, etc.).

We aim to represent a complicated function $f(x)$ by a simpler function $\phi(x; a_0, \ldots, a_n)$ where $a_0, \ldots, a_n$ are parameters chosen to give the best approximation of f.

Common ways to do this are:

1). Interpolation – choose the a_i so that
$\phi(x_i; a_0, \ldots, a_n) = f(x_i)$ on a prescribed set of points $\{x_i \mid i = 0, \ldots, n\}$. ← seen in 2H Num. An. (will study more)

2). Least-squares approximation – choose the a_i to minimise
$$\|f - \phi\|_2 := \left(\int_a^b (f(x) - \phi(x; a_0, \ldots, a_n))^2 \, dx \right)^{1/2}.$$ ← seen in 2H Num. An. (not in this course)

3). Minimax approximation – choose the a_i to minimise
$$\|f - \phi\|_\infty := \max_{x \in [a,b]} |f(x) - \phi(x; a_0, \ldots, a_n)|.$$ ← will study.

COURSE OUTLINE

0). Revision of polynomial interpolation.

1). Piecewise polynomial interpolation (splines).

2). Minimax approximation.

3). Trigonometric interpolation. (discrete Fourier transform).

0) Revision of Polynomial Interpolation. (2H Numerical Analysis).

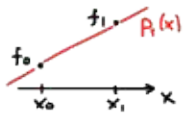
Main points:- (1) Given $f \in C[a,b]$ and $n+1$ distinct points $a \le x_0 < x_1 < \dots < x_n \le b$, there is a unique polynomial $p_n \in \mathbb{P}_n$ such that $p_n(x_i) = f(x_i)$ for $i = 0, \dots, n$.

($\mathbb{P}_n$: polynomials of degree $\le n$.)

(2) You have to be very careful where the points x_i are placed.

The proof of existence is by construction with Lagrange polynomials.

e.g. $n=1$



$$p_1(x) = \frac{x - x_1}{x_0 - x_1} f_0 + \frac{x - x_0}{x_1 - x_0} f_1.$$

In general,

$$p_n(x) = \sum_{i=0}^{n} f_i \, l_i(x) \quad \text{where} \quad l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$ are the Lagrange polynomials.

You should have seen the error formula:

Thm 0.1 – Let $f \in C^{n+1}[a,b]$ and $a \le x_0 < x_1 < \dots < x_n \le b$. Let $p_n \in \mathbb{P}_n$ be such that $p_n(x_i) = f(x_i)$ for $i = 0, \dots, n$. Then for every $x \in [a,b]$ there is a $\xi \in [a,b]$, depending on x, such that

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

where $$w_{n+1}(x) = \prod_{j=0}^{n} (x - x_j)$$ is the error polynomial.

- Disadvantage: can only apply this formula if f is smooth enough.
- Can use this to put an upper bound:

$$\| f - p_n \|_\infty \le \frac{\| w_{n+1} \|_\infty \, \| f^{(n+1)} \|_\infty}{(n+1)!}$$

The ∞-norm means $\| f \|_\infty = \max_{x \in [a,b]} |f(x)|$.

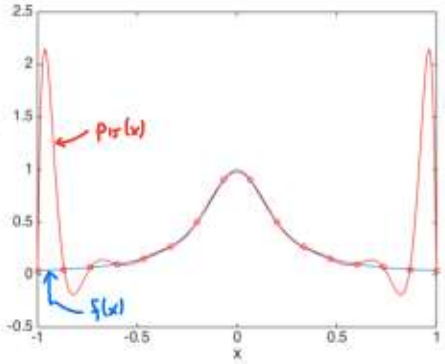
Suppose we take equally-spaced points $x_j = a + \left(\frac{b-a}{n}\right) j$ for $j = 0, \dots, n$. As we increase the resolution n, we would like $\| f - p_n \|_\infty \to 0$ as $n \to \infty$.

Problem:- The sequence $\{p_n\}$ may not converge to f because $\| w_{n+1} \|_\infty \| f^{(n+1)} \|_\infty$ may tend to ∞ faster than $(n+1)!$.

A famous example is due to Runge (1901):

Example: $f(x) = \frac{1}{1 + 25x^2}$ $x \in [-1, 1]$.

This function is well-behaved with all derivatives continuous and bounded, yet p_n diverges for equally-spaced points.



Remark:- Actually it does converge near $x = 0$. These kind of interpolants are used in higher-order (eg. 6th) finite difference methods for solving PDEs.

There are various ways to avoid the Runge phenomenon when approximating a function by polynomials:

1). choose the nodes carefully (revise today).
2). use splines ⟶ good if you aren't free to choose nodes.
3). use minimax or least-squares instead of interpolation.

0.1). CHEBYSHEV INTERPOLATION (still revision of 2H Numerical Analysis).

Recall that the error in polynomial interpolation satisfies

$$\|f - p_n\|_\infty \leq \frac{\|w_{n+1}\|_\infty \|f^{(n+1)}\|_\infty}{(n+1)!}$$

see that smoother functions will be easier to interpolate, in general.

We can't control $f^{(n+1)}$ (derivatives of the original function) but we can change the error polynomial

$$w_{n+1}(x) = \prod_{j=0}^{n} (x - x_j)$$

by moving the nodes x_j.

We will show that a good set of nodes are the Chebyshev nodes, defined as roots of the Chebyshev polynomial

$$T_n(x) = \cos\left(n \arccos(x)\right).$$

↖ from French transliteration "Tchebischeff" (1850s).

To see that this is a polynomial, let $\theta = \arccos(x)$, so

$$T_n(x) = \cos(n\theta) = \mathrm{Re}\left(e^{in\theta}\right) = \tfrac{1}{2}\left(e^{in\theta} + e^{-in\theta}\right) = \tfrac{1}{2}\left(z^n + z^{-n}\right) \quad \text{where } z = e^{i\theta}.$$

← complex number on unit circle

Now

$$\tfrac{1}{2}(z + z^{-1})(z^n + z^{-n}) = \tfrac{1}{2}\left(z^{n+1} + z^{-n-1}\right) + \tfrac{1}{2}\left(z^{n-1} + z^{-n+1}\right) \quad \text{for } n \geq 1$$

i.e.

$$2x\,T_n(x) = T_{n+1}(x) + T_{n-1}(x).$$

$$\Leftrightarrow \quad T_{n+1}(x) = 2x\,T_n(x) - T_{n-1}(x).$$

By induction, each $T_n(x)$ with $n \geq 1$ is a polynomial of degree exactly n, with leading coefficient 2^{n-1}.

The roots of $T_n(x)$ are

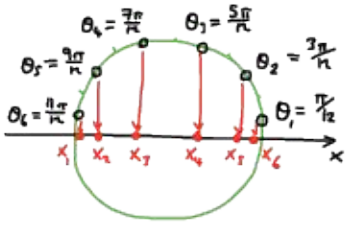
$$n\theta_j = \left(j - \tfrac{1}{2}\right)\pi \quad \text{for } j = 1, \dots, n$$

$$\Rightarrow \quad \tilde{x}_j = \cos\left(\pi - \frac{(j-\frac{1}{2})\pi}{n}\right) \quad \text{for } j = 1, \dots, n.$$

↑ reverse order

Notice that the nodes cluster near the end-points $x = \pm 1$.

e.g. $n = 6$



Lemma 0.2:- Suppose $p_n \in \mathcal{P}_n$ interpolates $f \in C^{(n+1)}[-1,1]$ at the Chebyshev nodes $\tilde{x}_0, \dots, \tilde{x}_n$. Let $\tilde{w}_{n+1}$ be the error polynomial for these nodes. Then

$$\tilde{w}_{n+1}(x) = \frac{1}{2^n} T_{n+1}(x).$$

Proof:- Since T_{n+1} and $\tilde{w}_{n+1}$ both belong to $\mathcal{P}_{n+1}$ and have $n+1$ roots (at $\tilde{x}_0, \dots, \tilde{x}_n$), there is a $c \in \mathbb{R}$ such that $T_{n+1} = c\,\tilde{w}_{n+1}$.

But the leading coefficient of T_{n+1} is 2^n and of $\tilde{w}_{n+1}$ is 1, so $c = 2^n$. ∎

Now we can show that the Chebyshev nodes are a good choice.

Thm 0.3 :- Suppose $p_n \in \mathcal{P}_n$ interpolates $f \in C^{n+1}[-1,1]$ at the Chebyshev nodes $\tilde{x}_0, \dots, \tilde{x}_n$. Let $\tilde{w}_{n+1}$ be the error polynomial for these nodes. Let $x_0, \dots, x_n$ be $n+1$ arbitrary, distinct nodes with error polynomial w_{n+1}. Then

$$\|\tilde{w}_{n+1}\|_\infty \leq \|w_{n+1}\|_\infty.$$

i.e. interpolating at Chebyshev nodes will minimise $\|w_{n+1}\|_\infty$. ← not the same as minimising $\|f - p_n\|_\infty$ since different x's are weighted by the $f^{(n+1)}(\xi)$ term. (later).

Proof :- Suppose there exist nodes $x_0, \dots, x_n$ such that $\|\tilde{w}_{n+1}\|_\infty > \|w_{n+1}\|_\infty$.

Then $q := \tilde{w}_{n+1} - w_{n+1}$ belongs to $\mathcal{P}_n$.

Now the extreme values of T_{n+1} occur at

$$\tilde{y}_j = \cos\left(\frac{j\pi}{n+1}\right) \qquad j = 0, \dots, n+1$$

and $T_{n+1}(\tilde{y}_j) = (-1)^j$.

Moreover, $\tilde{w}_{n+1} = \frac{1}{2^n} T_{n+1}$, so the maxima of $|\tilde{w}_{n+1}|$ occur at these points, with

$$\tilde{w}_{n+1}(y_j) = \tfrac{1}{2^n}(-1)^j.$$

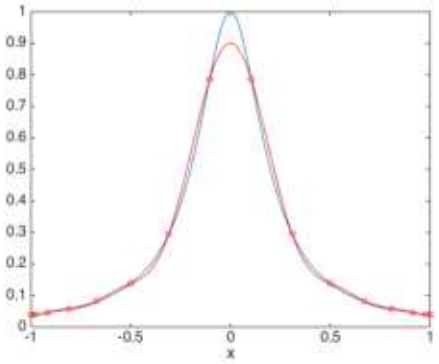
By assumption we must have $|w_{n+1}| < |\tilde{w}_{n+1}|$ at these points, so

$$\begin{cases} q(\tilde{y}_j) > 0 & \text{if } j \text{ even} \\ < 0 & \text{if } j \text{ odd.} \end{cases} \quad \} \; n+1 \text{ sign changes}$$

By the Intermediate Value Thm, q must have $n+1$ roots. Since $q \in \mathcal{P}_n$, we must have $q = 0$. ▨

Example:- $f(x) = \dfrac{1}{1+25x^2}$, $x \in [-1,1]$.

With Chebyshev points, the Runge phenomenon is absent.



Remark:- In fact, polynomial interpolants in Chebyshev points converge even if f is not in C^{n+1} – for example if f is only Lipschitz continuous (Trefethen).

But smoother f leads to more rapid convergence. (by Thm 0.1).

↳ e.g. $f(x) = |x|$.

There are various ways to avoid the Runge phenomenon when approximating a function by polynomials:

1). choose the nodes carefully (revise today).
2). use splines ⟶ good if you aren't free to choose nodes.
3). use minimax or least-squares instead of interpolation.

0.1). CHEBYSHEV INTERPOLATION (still revision of 2H Numerical Analysis).

Recall that the error in polynomial interpolation satisfies

$$\|f - p_n\|_\infty \leq \frac{\|w_{n+1}\|_\infty \|f^{(n+1)}\|_\infty}{(n+1)!}$$

see that smoother functions will be easier to interpolate, in general.

We can't control $f^{(n+1)}$ (derivatives of the original function) but we can change the error polynomial

$$w_{n+1}(x) = \prod_{j=0}^{n} (x - x_j)$$

by moving the nodes x_j.

We will show that a good set of nodes are the Chebyshev nodes, defined as roots of the Chebyshev polynomial

$$T_n(x) = \cos\left(n \arccos(x)\right).$$

↖ from French transliteration "Tchebischeff" (1850s).

To see that this is a polynomial, let $\theta = \arccos(x)$, so

$$T_n(x) = \cos(n\theta) = \mathrm{Re}\left(e^{in\theta}\right) = \tfrac{1}{2}\left(e^{in\theta} + e^{-in\theta}\right) = \tfrac{1}{2}\left(z^n + z^{-n}\right) \quad \text{where } z = e^{i\theta}.$$

← complex number on unit circle

Now

$$\tfrac{1}{2}(z + z^{-1})(z^n + z^{-n}) = \tfrac{1}{2}\left(z^{n+1} + z^{-n-1}\right) + \tfrac{1}{2}\left(z^{n-1} + z^{-n+1}\right) \quad \text{for } n \geq 1$$

i.e.

$$2x\,T_n(x) = T_{n+1}(x) + T_{n-1}(x).$$

$$\Leftrightarrow \quad T_{n+1}(x) = 2x\,T_n(x) - T_{n-1}(x).$$

By induction, each $T_n(x)$ with $n \geq 1$ is a polynomial of degree exactly n, with leading coefficient 2^{n-1}.

The roots of $T_n(x)$ are

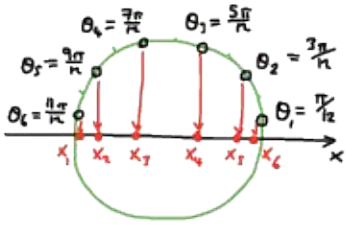
$$n\theta_j = \left(j - \tfrac{1}{2}\right)\pi \quad \text{for } j = 1, \dots, n$$

$$\Rightarrow \quad \tilde{x}_j = \cos\left(\pi - \frac{(j-\frac{1}{2})\pi}{n}\right) \quad \text{for } j = 1, \dots, n.$$

↑ reverse order

Notice that the nodes cluster near the end-points $x = \pm 1$.

e.g. $n = 6$



Lemma 0.2:- Suppose $p_n \in \mathcal{P}_n$ interpolates $f \in C^{(n+1)}[-1,1]$ at the Chebyshev nodes $\tilde{x}_0, \dots, \tilde{x}_n$. Let $\tilde{w}_{n+1}$ be the error polynomial for these nodes. Then

$$\tilde{w}_{n+1}(x) = \frac{1}{2^n} T_{n+1}(x).$$

Proof:- Since T_{n+1} and $\tilde{w}_{n+1}$ both belong to $\mathcal{P}_{n+1}$ and have $n+1$ roots (at $\tilde{x}_0, \dots, \tilde{x}_n$), there is a $c \in \mathbb{R}$ such that $T_{n+1} = c\,\tilde{w}_{n+1}$.

But the leading coefficient of T_{n+1} is 2^n and of $\tilde{w}_{n+1}$ is 1, so $c = 2^n$. ∎

Now we can show that the Chebyshev nodes are a good choice.

Thm 0.3 :- Suppose $p_n \in \mathcal{P}_n$ interpolates $f \in C^{n+1}[-1,1]$ at the Chebyshev nodes $\tilde{x}_0, \dots, \tilde{x}_n$. Let $\tilde{w}_{n+1}$ be the error polynomial for these nodes. Let $x_0, \dots, x_n$ be $n+1$ arbitrary, distinct nodes with error polynomial w_{n+1}. Then

$$\|\tilde{w}_{n+1}\|_\infty \leq \|w_{n+1}\|_\infty.$$

i.e. interpolating at Chebyshev nodes will minimise $\|w_{n+1}\|_\infty$. ← not the same as minimising $\|f - p_n\|_\infty$ since different x's are weighted by the $f^{(n+1)}(\xi)$ term. (later).

Proof :- Suppose there exist nodes $x_0, \dots, x_n$ such that $\|\tilde{w}_{n+1}\|_\infty > \|w_{n+1}\|_\infty$.

Then $q := \tilde{w}_{n+1} - w_{n+1}$ belongs to $\mathcal{P}_n$.

Now the extreme values of T_{n+1} occur at

$$\tilde{y}_j = \cos\left(\frac{j\pi}{n+1}\right) \qquad j = 0, \dots, n+1$$

and $T_{n+1}(\tilde{y}_j) = (-1)^j$.

Moreover, $\tilde{w}_{n+1} = \frac{1}{2^n} T_{n+1}$, so the maxima of $|\tilde{w}_{n+1}|$ occur at these points, with

$$\tilde{w}_{n+1}(y_j) = \tfrac{1}{2^n}(-1)^j.$$

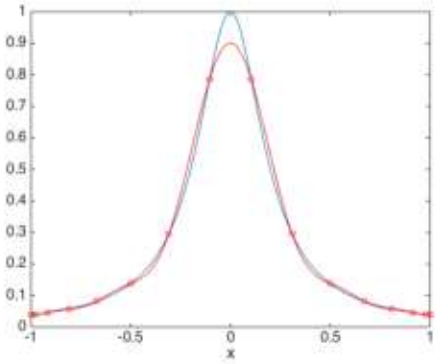
By assumption we must have $|w_{n+1}| < |\tilde{w}_{n+1}|$ at these points, so

$$\begin{cases} q(\tilde{y}_j) > 0 & \text{if } j \text{ even} \\ < 0 & \text{if } j \text{ odd.} \end{cases} \quad \} \; n+1 \text{ sign changes}$$

By the Intermediate Value Thm, q must have $n+1$ roots. Since $q \in \mathcal{P}_n$, we must have $q = 0$. ▨

Example:- $f(x) = \dfrac{1}{1+25x^2}$, $x \in [-1,1]$.

With Chebyshev points, the Runge phenomenon is absent.



Remark:- In fact, polynomial interpolants in Chebyshev points converge even if f is not in C^{n+1} – for example if f is only Lipschitz continuous (Trefethen).

But smoother f leads to more rapid convergence. (by Thm 0.1).

↳ e.g. $f(x) = |x|$.

1). PIECEWISE POLYNOMIAL INTERPOLATION

Key idea:- Avoid oscillations by using lots of low-degree polynomial interpolants.
- useful if you aren't free to select the points, but sacrifices smoothness.

Let $s(x)$ be a function which interpolates $f \in C[a,b]$ at the knots $x_0 < x_1 < \dots < x_n$, consisting of n polynomials $s_i(x)$ in each interval $[x_i, x_{i+1}]$. Then $s(x)$ is called a spline of degree N if it is $N-1$ times continuously differentiable.

This means that the pieces satisfy the following matching conditions at the interior knots:

automatic from interpolation requirement

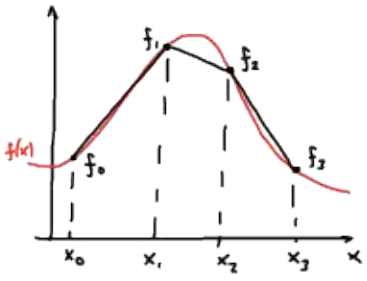
$$\left.\begin{aligned} s_{i-1}(x_i) &= s_i(x_i) \\ s'_{i-1}(x_i) &= s'_i(x_i) \\ &\vdots \\ s^{(N-1)}_{i-1}(x_i) &= s^{(N-1)}_i(x_i). \end{aligned}\right\} \text{ for } i = 1, \dots, n-1.$$

In other words, splines have a certain degree of smoothness.

Remark:-
- Splines are useful in computer aided design (e.g. car or aeroplane bodies).
- Originally a draftsman's "spline" was a strip of flexible wood/metal held by pegs and traced by hand to produce a smooth curve. → will see cubic splines later

1.1). LINEAR SPLINES ($N = 1$).

This is simply piecewise-linear interpolation, with no derivative conditions.



The requirements are just

$$\begin{aligned} s_0(x_0) &= f_0, & s_0(x_1) &= f_1 \\ s_1(x_1) &= f_1, & s_1(x_2) &= f_2 \\ &\vdots & &\vdots \\ s_{n-1}(x_{n-1}) &= f_{n-1}, & s_{n-1}(x_n) &= f_n. \end{aligned}$$

This uniquely defines each $s_i(x)$ by

$$s_i(x) = f_i + \left(\frac{f_{i+1} - f_i}{x_{i+1} - x_i}\right)(x - x_i)$$

and hence

$$s(x) = \begin{cases} s_0(x), & x_0 \le x \le x_1, \\ \vdots \\ s_{n-1}(x), & x_{n-1} \le x \le x_n. \end{cases}$$

By taking more and more points, functions in $C^2[a,b]$ can be approximated arbitrarily well by linear splines...

Thm 1.1 :- Let $f \in C^1[a,b]$ and $s(x)$ be a linear spline that interpolates f at the knots $x_0 < \dots < x_n$. If $h = \max_{1 \le i \le n} (x_i - x_{i-1})$ then

$$\|f - s\|_\infty \le \tfrac{1}{8} h^2 \|f''\|_\infty.$$

Proof:- In each interval $[x_i, x_{i+1}]$, s is a linear interpolant, so by Thm 0.1,

$$f(x) - s_i(x) = \frac{w_2(x) f''(\xi)}{2} \quad \text{for some } \xi \in [x_i, x_{i+1}].$$
$$= (x-x_i)(x-x_{i+1}) \frac{f''(\xi)}{2}.$$

So

$$|f(x) - s_i(x)| \leq |(x-x_i)(x-x_{i+1})| \frac{|f''(\xi)|}{2} = (x-x_i)(x_{i+1}-x) \frac{|f''(\xi)|}{2}$$

by symmetry, maximum is at mid-point $x = \frac{x_i + x_{i+1}}{2}$.

$$\leq \left(\frac{x_i+x_{i+1}}{2} - x_i\right)\left(x_{i+1} - \frac{x_i+x_{i+1}}{2}\right) \frac{|f''(\xi)|}{2}$$
$$= \frac{1}{4}(x_{i+1}-x_i)^2 \frac{|f''(\xi)|}{2}.$$

Hence

$$\|f - s\|_\infty \leq \frac{1}{8} h^2 \|f''\|_\infty. \qquad \blacksquare$$

Clearly to minimize error we should add more knots where the function has largest second derivative.

see this in the sketch above.

Linear splines are the "flattest" among all functions interpolating f at a given set of knots, in the sense that the average slope $\|s'\|_2$ is the smallest.

$\|s'\|_2 = \left(\int_a^b |s'(x)|^2 dx\right)^{1/2}$.

Thm 1.2 :- Let s be a linear spline interpolating $f \in C[a,b]$ at the knots $x_0 < \ldots < x_n$. Then

$$\|s'\|_2 \leq \|f'\|_2.$$

← dates back to J. C. Holladay in 1950s.

Proof:- Note that

$$\|f' - s'\|_2^2 = \int_a^b |f' - s'|^2 dx$$
$$= \int_a^b \{(f')^2 - 2f's' + (s')^2\} dx$$
$$= \|f'\|_2^2 + \|s'\|_2^2 - 2\int_a^b f's' dx$$
$$= \|f'\|_2^2 - \|s'\|_2^2 - 2\int_a^b (f' - s')s' dx.$$

so

$$\|f'\|_2^2 = \|f' - s'\|_2^2 + \|s'\|_2^2 + 2\underbrace{\int_a^b (f' - s')s' dx}_{\circledast}.$$

If $\circledast$ vanishes then we are done.

Integrate by parts:

$$\circledast = \int_a^b \{(f-s)s'\}' dx - \int_a^b (f-s)s'' dx.$$

split into separate intervals

$$= \sum_{i=1}^n \int_{x_i}^{x_{i+1}} [(f-s_i)s_i']' dx - \sum_{i=1}^n \int_{x_i}^{x_{i+1}} (f-s_i)\underbrace{s_i''}_{=0} dx$$
$$= \sum_{i=1}^n \left\{(f-s_i)s_i'\Big|_{x=x_{i+1}} - (f-s_i)s_i'\Big|_{x=x_i}\right\}$$
$$= 0 \quad \text{because } s_i(x_i) = f(x_i) \text{ and } s_i(x_{i+1}) = f(x_{i+1}). \qquad \blacksquare$$

Quadratic splines are less used in practice (no nice variational property), so we move to

1.2) CUBIC SPLINES

A cubic spline ($N = 3$) is made up of cubic pieces

$$s_i(x) = a_i + b_i x + c_i x^2 + d_i x^3, \qquad i = 0, \dots, n-1,$$

that satisfy the following conditions:

1). Interpolation at the knots

$$\left.\begin{aligned} s_i(x_i) &= f_i \\ s_i(x_{i+1}) &= f_{i+1} \end{aligned}\right\} \text{ for } i = 0, \dots, n-1. \longrightarrow 2n \text{ conditions}$$

2). Matching first derivatives at interior knots:

$$s'_{i-1}(x_i) = s'_i(x_i) \quad \text{for } i = 1, \dots, n-1. \longrightarrow n-1 \text{ conditions}$$

3). Matching second derivatives at interior knots:

$$s''_{i-1}(x_i) = s''_i(x_i) \quad \text{for } i = 1, \dots, n-1. \longrightarrow n-1 \text{ conditions}$$

We have $4n$ unknowns (a_i, b_i, c_i, d_i) but only $4n-2$ conditions, so we need 2 extra conditions, whatever the value of n. → so we can't impose continuity of s''' at every knot.

There are several ways of choosing the extra two conditions – some popular ones are:

- Natural cubic spline: $s''_0(x_0) = s''_{n-1}(x_n) = 0.$

→ Like the physical spline: ends are straight.

- Complete/clamped cubic spline: $\begin{cases} s'_0(x_0) = f'(x_0) \\ s'_{n-1}(x_n) = f'(x_n) \end{cases}$ ← needs extra info about f.

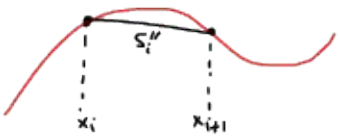
- Not-a-knot cubic spline: $\begin{cases} s'''_0(x_1) = s'''_1(x_1) \\ s'''_{n-2}(x_{n-1}) = s'''_{n-1}(x_{n-1}) \end{cases}$

Unlike the linear spline, the different s_i are now coupled and cannot be computed independently. We will show how to formulate a linear system (will also let us decide existence and uniqueness). → makes cubic splines convenient.

Let $M_i = s''_i(x_i)$ — called the moments of the cubic spline.

Since s_i is cubic, s''_i is linear, so in $[x_i, x_{i+1}]$ we can write

$$s''_i(x) = \left(\frac{x_{i+1} - x}{h_i}\right) M_i + \left(\frac{x - x_i}{h_i}\right) M_{i+1}, \quad \text{where } h_i = x_{i+1} - x_i.$$



Note that we have imposed the condition that the second derivative is continuous.

Now integrate twice:

$$s_i(x) = \frac{(x_{i+1} - x)^3}{6h_i} M_i + \frac{(x - x_i)^3}{6h_i} M_{i+1} + \alpha_i (x - x_i) + \beta_i (x_{i+1} - x).$$

To determine α_i and β_i, we can apply the interpolation conditions:

$$f_i = s_i(x_i) = \frac{1}{6}h_i^2 M_i + h_i\beta_i \qquad \Rightarrow \quad \beta_i = \frac{f_i - \frac{1}{6}h_i^2 M_i}{h_i}$$
$$f_{i+1} = s_i(x_{i+1}) = \frac{1}{6}h_i^2 M_{i+1} + h_i\alpha_i \qquad \alpha_i = \frac{f_{i+1} - \frac{1}{6}h_i^2 M_{i+1}}{h_i}.$$

We still have to apply the first derivative condition. For each piece, we have

$$s_i'(x) = -\frac{(x_{i+1}-x)^2}{2h_i}M_i + \frac{(x-x_i)^2}{2h_i}M_{i+1} + \alpha_i - \beta_i$$
$$= -\frac{(x_{i+1}-x)^2}{2h_i}M_i + \frac{(x-x_i)^2}{2h_i}M_{i+1} + \frac{1}{h_i}(f_{i+1}-f_i) + \frac{1}{6}h_i(M_i - M_{i+1}).$$

Changing the index gives

$$s_{i-1}'(x) = -\frac{(x_i-x)^2}{2h_{i-1}}M_{i-1} + \frac{(x-x_{i-1})^2}{2h_{i-1}}M_i + \frac{1}{h_{i-1}}(f_i - f_{i-1}) + \frac{1}{6}h_{i-1}(M_{i-1} - M_i).$$

So

$$s_{i-1}'(x_i) = s_i'(x_i)$$
$$\Leftrightarrow \quad \frac{h_{i-1}^2}{2h_{i-1}}M_i + \frac{1}{h_{i-1}}(f_i - f_{i-1}) + \frac{1}{6}h_{i-1}(M_{i-1} - M_i) = -\frac{h_i^2}{2h_i}M_i + \frac{1}{h_i}(f_{i+1}-f_i) + \frac{1}{6}h_i(M_i - M_{i+1})$$
$$\Leftrightarrow \quad h_{i-1}M_{i-1} + 2(h_{i-1}+h_i)M_i + h_iM_{i+1} = \underbrace{6\left(\frac{f_{i+1}-f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}}\right)}_{b_i} \quad \text{for } i = 1, \dots, n-1.$$

Together with the natural spline boundary conditions $M_0 = M_n = 0$, this gives a tridiagonal system of linear equations,

$$\begin{bmatrix} 1 & 0 & 0 & & & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & & \\ 0 & h_1 & 2(h_1+h_2) & h_2 & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & & 2(h_{n-2}+h_{n-1}) & h_{n-1} & \\ 0 & & & 0 & 1 & \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} 0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ 0 \end{bmatrix}$$

The system is diagonally dominant since $2(h_{i-1}+h_i) > h_{i-1}+h_i$ so has a unique solution.

[A matrix A is strictly diagonally dominant if $|a_{ii}| > \sum_{j\neq i}|a_{ij}|$ for all rows i.

Proof that every s.d.d. matrix is nonsingular:

Suppose not. Then there exists a vector $\vec{u} \neq 0$ such that $A\vec{u} = \vec{0}$, and $\vec{u}$ has some element u_i of largest magnitude.

We have

$$\sum_j a_{ij}u_j = 0 \quad \Leftrightarrow \quad a_{ii}u_i = -\sum_{j\neq i} a_{ij}u_j$$
$$\Leftrightarrow \quad a_{ii} = -\sum_{j\neq i}\left(\frac{u_j}{u_i}\right)a_{ij}$$
$$\Leftrightarrow \quad |a_{ii}| \leq \sum_{j\neq i}\underbrace{\left|\frac{u_j}{u_i}\right|}_{\leq 1}|a_{ij}| \leq \sum_{j\neq i}|a_{ij}| \quad \text{— contradiction.} \ \blacksquare$$

Example: Fit a natural cubic spline through the data $(0,0)$, $(1,1)$, $(2,8)$.

Here $x_0 = 0$, $x_1 = 1$, $x_2 = 2$, $h_0 = h_1 = 1$.

In $[0,1]$:

$$S_0(x) = \frac{(1-x)^3}{6} M_0 + \frac{x^3}{6} M_1 + \alpha_0 x + \beta_0 (1-x)$$

$M_0 = 0$ (natural c. spl.)

$$= \frac{x^3}{6} M_1 + \alpha_0 x + \beta_0 (1-x)$$

and

$$f_0 = S_0(x_0) \Rightarrow 0 = \beta_0$$

$$f_1 = S_0(x_1) \Rightarrow 1 = \frac{1}{6} M_1 + \alpha_0 \Rightarrow S_0(x) = \frac{x^3}{6} M_1 + \left(1 - \frac{1}{6} M_1\right) x.$$

In $[1,2]$:

$$S_1(x) = \frac{(2-x)^3}{6} M_1 + \frac{(x-1)^3}{6} M_2 + \alpha_1 (x-1) + \beta_1 (2-x)$$

$M_2 = 0$

$$= \frac{(2-x)^3}{6} M_1 + \alpha_1 (x-1) + \beta_1 (2-x)$$

and

$$f_1 = S_1(x_1) \Rightarrow 1 = \frac{1}{6} M_1 + \beta_1$$

$$f_2 = S_1(x_2) \Rightarrow 8 = \alpha_1 \Rightarrow S_1(x) = \frac{(2-x)^3}{6} M_1 + 8(x-1) + \left(1 - \frac{1}{6} M_1\right)(2-x).$$

Finally the first derivative condition gives

$$S_0'(x_1) = S_1'(x_1)$$

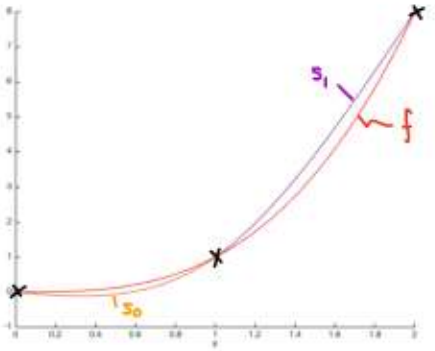
$$\Leftrightarrow \frac{3x_1^2}{6} M_1 + \left(1 - \frac{1}{6} M_1\right) x_1 = -\frac{(2-x_1)^2}{2} M_1 + 8 - 1 + \frac{1}{6} M_1$$

$$\Leftrightarrow \frac{1}{2} M_1 + 1 - \frac{1}{6} M_1 = -\frac{1}{2} M_1 + 7 + \frac{1}{6} M_1$$

$$\Leftrightarrow \frac{2}{3} M_1 = 6 \Leftrightarrow M_1 = 9.$$

So our natural cubic spline is

$$s(x) = \begin{cases} \frac{3}{2} x^3 - \frac{1}{2} x & \text{for } 0 \le x \le 1, \\ \frac{3}{2}(2-x)^3 + 8(x-1) - \frac{1}{2}(2-x) & \text{for } 1 \le x \le 2. \end{cases}$$



Observe that our spline is not $s(x) = x^3$, even though the data come from $f(x) = x^3$, i.e. natural cubic splines do not reproduce cubic functions.

↖ unlike linear splines.

this leads to a lower order of convergence → than e.g. complete c. spl.

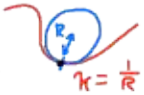
This is because $f(x) = x^3$ does not satisfy the end conditions $f''(x_0) = f''(x_n) = 0$.

Thm 1.3. (Holladay) Let s be the natural cubic spline interpolating $f \in C^2[a,b]$ at the knots $x_0 < \dots < x_n$. Then
$$\|s''\|_2 \leq \|f''\|_2 .$$

- i.e. s gives the smallest value of the integral $\left(\int_a^b |s''(x)|^2 dx\right)^{1/2}$ over the class of admissible functions: it is the "smoothest" interpolant. ← cf. Thm 1.2 for linear splines.

- Physical interpretation:-

The local curvature of a graph $y = f(x)$ is $\kappa = \dfrac{y''}{(1+|y'|^2)^{3/2}}$.



$\kappa = \frac{1}{R}$ where R = radius of curvature

The total internal strain energy stored in an elastic beam is proportional to $\int \kappa^2 dx$, so if $|y'|$ is small, the shape of minimum energy will minimise $\int (y'')^2 dx$, ie. will be a cubic spline. → so this is close to the shape taken by a drafting spline.

Proof of Thm 1.3:- As in Thm 1.2, start with

$$\begin{aligned}
\|f'' - s''\|_2^2 &= \int_a^b |f'' - s''|^2 dx \\
&= \int_a^b \{(f'')^2 - 2f''s'' + (s'')^2\} dx \\
&= \|f''\|_1^2 + \|s''\|_2^2 - 2\int_a^b f''s'' dx \\
&= \|f''\|_2^2 - \|s''\|_1^2 - 2\int_a^b (f'' - s'')s'' dx .
\end{aligned}$$

Our task is to show that the last term vanishes. ← Same idea as Thm 1.2 !

$$\begin{aligned}
\int_a^b (f'' - s'')s'' dx &= \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} (f'' - s_i'')s_i'' dx \\
&= \underbrace{\sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} [(f' - s_i')s_i'']' dx}_{①} - \underbrace{\sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} (f' - s_i')s_i''' dx}_{②}
\end{aligned}$$

(s_i''' // A_i (constant for each s_i).)

Now
$$① = \sum_{i=0}^{n-1} \underbrace{(f'(x) - s_i'(x))}_{=0 \text{ at interior nodes}} s_i''(x)\Big|_{x_i}^{x_{i+1}} = (f'(x_n) - s_{n-1}'(x_n))s_{n-1}''(x_n) - (f'(x_0) - s_0'(x_0))s_0''(x_0)$$
$$= 0 \quad \text{for natural cubic spline } (s_{n-1}''(x_n) = s_0''(x_0) = 0)$$

leaving
$$② = \sum_{i=0}^{n-1} A_i \int_{x_i}^{x_{i+1}} (f - s_i)' dx$$
$$= \sum_{i=0}^{n-1} A_i \{(f(x_{i+1}) - s_i(x_{i+1})) - (f(x_i) - s_i(x_i))\} \overset{\text{by interpolation conditions}}{=} 0.$$

∎

In fact, the natural cubic spline is the only function that achieves $\|f'' - s''\|_2^2 = 0$:-
Since the integrand in $\int_a^b (f'' - s'')^2 dx$ is non-negative and continuous, we must have
$$(f'' - s'')^2 = 0 \iff f'' = s''$$
$$\iff f(x) = s(x) + cx + d.$$
The end conditions $s(x_0) = f(x_0)$ and $s(x_n) = f(x_n)$ show that $c = d = 0$, so only the natural cubic spline achieves $\|f''\|_2 = \|s''\|_2$.

1.3). B-SPLINES

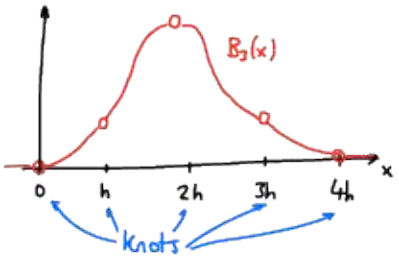
An alternative approach for computing cubic splines is to use basis functions called B-splines which are themselves cubic splines but with compact support.

B for "Basis" (coined by Schoenberg in 1946).

For simplicity, assume constant knot spacing h.

The cubic B-spline centered at $x = 2h$ has the form

$$B_3(x) = \begin{cases} x^3 = b_1(x) & \text{for } 0 \le x \le h, \\ x^3 - 4(x-h)^3 = b_2(x) & \text{for } h \le x \le 2h, \\ x^3 - 4(x-h)^3 + 6(x-2h)^3 = b_3(x) & \text{for } 2h \le x \le 3h, \\ (4h-x)^3 = b_4(x) & \text{for } 3h \le x \le 4h, \\ 0 & \text{elsewhere.} \end{cases}$$



We can check that this is really a cubic spline:

i) Continuity of B_3 at knots:

$b_1(0) = 0$, $b_1(h) = h^3$, $b_2(2h) = 4h^3$, $b_3(3h) = h^3$, $b_4(4h) = 0$

$b_2(h) = h^3$ $b_3(2h) = 4h^3$ $b_4(3h) = h^3$

ii). Continuity of B_3' at the knots:

$b_1'(0) = 0$, $b_1'(h) = 3h^2$ $b_2'(2h) = 12h^2 - 12h^2 = 0$ $b_3'(3h) = 27h^2 - 48h^2 + 18h^2 = -3h^2$ $b_4'(4h) = 0$.

$b_2'(h) = 3h^2$ $b_3'(2h) = 0$ $b_4'(3h) = -3h^2$

iii). Continuity of B_3'' at the knots:

$b_1''(0) = 0$, $b_1''(h) = 6h$, $b_2''(2h) = 12h - 24h = -12h$ $b_3''(3h) = 18h - 48h + 36h = 6h$ $b_4''(4h) = 0$.

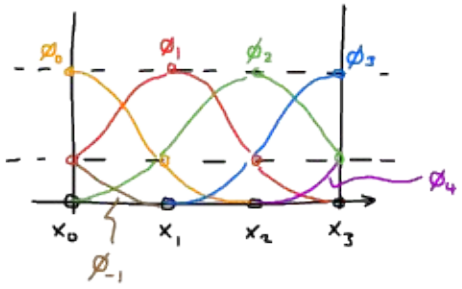
$b_2''(h) = 6h$, $b_3''(2h) = -12h$ $b_4''(3h) = 6h$

Hence $B_3 \in C^2(-\infty, \infty)$ and so is a cubic spline.

The idea is that any cubic spline can be written as a linear combination

$$s(x) = \sum_{i=-1}^{n+1} \alpha_i \phi_i(x) \qquad \text{where} \quad \phi_i(x) = \frac{1}{h^3} B_3(x - x_{i-2}).$$

e.g. $n = 3$



The extra basis functions ϕ_{-1}, ϕ_{n+1} are needed to incorporate boundary conditions.

Example:- Fit a natural cubic spline through the data $(0,0)$, $(1,1)$, $(2,8)$.

Here $h = 1$ and $n = 2$, so we need $\phi_{-1}, \phi_0, \phi_1, \phi_2, \phi_3$. We have $x_0 = 0$, $x_1 = 1$, $x_2 = 2$ and $x_{-1} = -1$, $x_{-2} = -2$, $x_{-3} = -3$, ...

The basis functions are: ↓ look at the picture!

$$\phi_{-1}(x) = B_3(x - x_{-3}) = \begin{cases} (4-x-3)^3 = (1-x)^3 & \text{in } [0,1] \\ 0 & \text{in } [0,2] \end{cases}$$

$$\phi_0(x) = B_3(x - x_{-2}) = \begin{cases} (x+2)^3 - 4(x+1)^3 + 6x^3 & \text{in } [0,1] \\ (2-x)^3 & \text{in } [1,2] \end{cases}$$

$$\phi_1(x) = B_3(x - x_{-1}) = \begin{cases} (x+1)^3 - 4x^3 & \text{in } [0,1] \\ (x+1)^3 - 4x^3 + 6(x-1)^3 & \text{in } [1,2] \end{cases}$$

$$\phi_2(x) = B_3(x - x_0) = \begin{cases} x^3 & \text{in } [0,1] \\ x^3 - 4(x-1)^3 & \text{in } [1,2]. \end{cases}$$

$$\phi_3(x) = B_3(x - x_1) = \begin{cases} 0 & \text{in } [0,1] \\ (x-1)^3 & \text{in } [1,2]. \end{cases}$$

In fact, to find the coefficients α_i, we just need to know $\phi_i(x_j)$ at the knots x_j.
The interpolation conditions give

$$\alpha_{-1}\phi_{-1}(x_0) + \alpha_0\phi_0(x_0) + \alpha_1\phi_1(x_0) + \alpha_2\phi_2(x_0) + \alpha_3\phi_3(x_0) = f_0 = 0.$$
(1, 4, 1)

$$\alpha_{-1}\phi_{-1}(x_1) + \alpha_0\phi_0(x_1) + \alpha_1\phi_1(x_1) + \alpha_2\phi_2(x_1) + \alpha_3\phi_3(x_1) = f_1 = 1$$
(1, 4, 1) ← notice the system is tridiagonal

$$\alpha_{-1}\phi_{-1}(x_2) + \alpha_0\phi_0(x_2) + \alpha_1\phi_1(x_2) + \alpha_2\phi_2(x_2) + \alpha_3\phi_3(x_2) = f_2 = 8.$$
(1, 4, 1)

To close the system we have the natural boundary conditions:

$$s''(x_0) = 0. \Rightarrow \alpha_{-1}\phi_{-1}''(x_0) + \alpha_0\phi_0''(x_0) + \alpha_1\phi_1''(x_0) = 0$$
(6, −12, 6) → from above

$$s''(x_3) = 0 \Rightarrow \alpha_1\phi_1''(x_3) + \alpha_2\phi_2''(x_3) + \alpha_3\phi_3''(x_3) = 0.$$
(6, −12, 6)

So the linear system is

$$\begin{bmatrix} 6 & -12 & 6 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 1 & 4 & 1 \\ 0 & 0 & 6 & -12 & 6 \end{bmatrix} \begin{bmatrix} \alpha_{-1} \\ \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 8 \\ 0 \end{bmatrix} \Rightarrow \begin{array}{l} \alpha_{-1} = \frac{1}{12} \\ \alpha_0 = 0 \\ \alpha_1 = -\frac{1}{12} \\ \alpha_2 = \frac{4}{3} \\ \alpha_3 = \frac{11}{4}. \end{array}$$

Thus

$$s(x) = \begin{cases} \frac{1}{12}(1-x)^3 - \frac{1}{12}(x+1)^3 + \frac{1}{3}x^3 + \frac{4}{3}x^3 = \frac{7}{2}x^3 - \frac{1}{2}x & \text{in } [0,1] \\ -\frac{1}{12}(x+1)^3 + \frac{1}{3}x^3 - \frac{1}{2}(x-1)^3 + \frac{4}{3}x^3 - \frac{14}{3}(x-1)^3 + \frac{11}{4}(x-1)^3 = -\frac{3}{2}x^3 + 9x^2 - \frac{19}{2}x + 3 & \text{in } [1,2] \end{cases}$$

↙ agrees with our answer in previous lecture.

Why are B-splines important?

- There is a general formula defining them for different orders of spline. [see Problem sheet].
- There is a fast algorithm for evaluating them (de Boor, 1972 — see SIAM article).

They are widely used in computer design (e.g. cars, aircraft) as well as for interpolating data.

2). Minimax Approximation

2.1). Weierstrass Approximation Theorem

Thm 2.1 (Weierstrass, 1885) — Let $f \in C[0,1]$. Then for any $\varepsilon > 0$ there exists a polynomial p such that $\|f - p\|_\infty < \varepsilon$.

→ aged 70

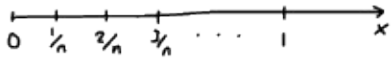
- For any other interval $[a,b]$ we can just change variables.
- This is stronger than our previous error bounds (e.g. Taylor's Thm or Lagrange interpolation) because it doesn't require differentiability of f, only continuity.

We will give Sergei Bernstein's proof (ca. 1910) which provides an explicit sequence of converging polynomials.

The n^{th} Bernstein polynomial for f is defined as

$$B_n(f,x) = \sum_{k=0}^{n} f\left(\tfrac{k}{n}\right)\binom{n}{k} x^k (1-x)^{n-k}.$$

usual Binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

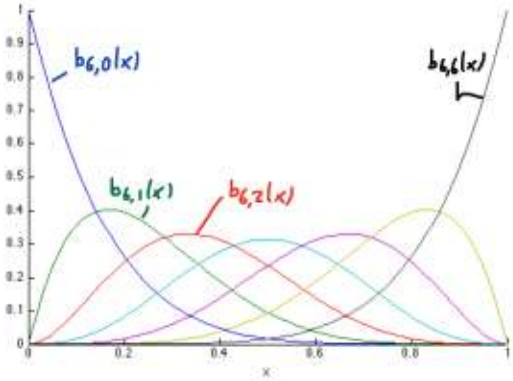


Note that

$$\sum_{k=0}^{n}\binom{n}{k} x^k (1-x)^{n-k} \overset{\text{binomial thm.}}{=} (x + (1-x))^n = 1^n = 1$$

so $B_n(f,x)$ at any point $x \in [0,1]$ is a weighted average of the $n+1$ function values $f(k/n)$.

e.g. the functions $b_{n,k}(x) := \binom{n}{k} x^k (1-x)^{n-k}$ for $n = 6$:



Notice that each $b_{n,k}(x)$ peaks near a different point of $[0,1]$.

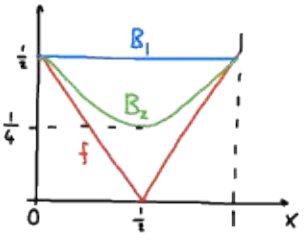
Example: Approximate $f(x) = |x - \tfrac{1}{2}|$ on $[0,1]$ with Bernstein polynomials.

$n=1$: $B_1(f,x) = f(0)\binom{1}{0}(1-x) + f(1)\binom{1}{1}x = \frac{1}{2}(1-x) + \frac{1}{2}x = \frac{1}{2}$.

$n=2$: $B_2(f,x) = f(0)\binom{0}{2}(1-x)^2 + f(\tfrac{1}{2})\binom{2}{1}x(1-x) + f(1)\binom{2}{2}x^2$

$= \frac{1}{2} - x + x^2$.

Convergence is slow ($\approx n^{-1/2}$) at $x = \frac{1}{2}$.



This slow convergence makes Bernstein approximation impractical.

Key idea of Bernstein's proof :- the weights $b_{n,k}(x)$ become more and more localised around $x = k/n$ as $n \to \infty$.

We will use the fact that $b_{n,k}(x) = \binom{n}{k} x^k (1-x)^{n-k}$ is the probability mass function of the binomial distribution — ie. the probability of getting k successes in n trials, each with probability x. ← recall from first year!

— the mean per trial is $E(k/n) = x$ and the variance is $Var(k/n) = \frac{x(1-x)}{n}$.

Since $Var(k/n)$ shrinks as n grows, each distribution clusters more tightly around $x = k/n$, and $B_n(f, k/n) \to f(k/n)$.

Proof of Thm 2.1 —

The error is

$$|f(x) - B_n(f,x)| = \left| \underbrace{\sum_{k=0}^{n} b_{n,k}(x)}_{=1} f(x) - \sum_{k=0}^{n} \binom{n}{k} x^k (1-x)^{n-k} f(k/n) \right|$$

$$\leq \sum_{k=0}^{n} |f(x) - f(k/n)| \, b_{n,k}(x).$$

Now divide the sum into k/n near x (major contribution), say $|x - k/n| \leq \delta$, and those with $|x - k/n| > \delta$.

The first part is

$$\sum_{|x - k/n| \leq \delta} |f(x) - f(k/n)| \, b_{n,k}(x).$$

We simply choose δ small enough so that $|f(x) - f(k/n)| < \varepsilon/2$ whenever $|x - k/n| \leq \delta$.

The second sum is

$$\sum_{|x - k/n| > \delta} |f(x) - f(k/n)| \, b_{n,k}(x) \leq M \underbrace{\sum_{|x - k/n| > \delta} b_{n,k}(x)}_{S} \qquad \text{where } M = \| f(x) - f(k/n) \|_\infty .$$

We hope that for n large enough we will have $S \leq \frac{\varepsilon}{2M}$. This is reasonable since $b_{n,k}(x)$ is sizeable over a narrower domain as n increases.

To prove it rigorously, we can use

Chebyshev's Inequality — If X is a discrete random variable with mean $E(x)$ and variance $Var(x)$, then

$$\text{Prob}\{ |X - E(x)| \geq s\sqrt{Var(x)} \} \leq \frac{1}{s^2}.$$

Here $E(k/n) = x$, $Var(k/n) = \frac{x(1-x)}{n}$, and S is the probability of being more than δ from the mean.

To ensure $S \leq \frac{\varepsilon}{2M}$, Chebyshev would do it if $\frac{1}{s^2} = \frac{\varepsilon}{2M}$, so

$$s\sqrt{Var(x)} \leq \delta \iff \delta \geq \sqrt{\frac{x(1-x)}{n} \frac{2M}{\varepsilon}} \iff n \geq \frac{2Mx(1-x)}{\varepsilon \delta^2}.$$

Thus it works for n large enough. ▨

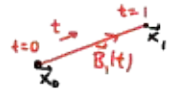
2.2). BEZIER CURVES

Bernstein polynomials also appear in the parametric formulae for Bézier curves.

Recall: $B_n(f,x) = \sum_{k=0}^{n} f(\frac{k}{n}) \binom{n}{k} x^k (1-x)^{n-k}$.

If we replace $f(\frac{k}{n})$ by positions $\vec{x}_k \in \mathbb{R}^2$ and x by t we get parametric expressions for Bézier curves:

e.g. $\vec{B}_1(t) = (1-t)\vec{x}_0 + t\vec{x}_1$.

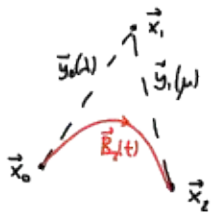


This is just linear interpolation, $\vec{B}_1(0) = \vec{x}_0$, $\vec{B}_1(1) = \vec{x}_1$.

e.g. $\vec{B}_2(t) = (1-t)^2\vec{x}_0 + 2(1-t)t\vec{x}_1 + t^2\vec{x}_2$.

When $t=0$, $\vec{B}_2(0) = \vec{x}_0$

$t=1$, $\vec{B}_2(1) = \vec{x}_2$



Claim: The tangents at $\vec{x}_0$ and $\vec{x}_2$ intersect at $\vec{x}_1$.

To see this, differentiate w.r.t. t:

$\vec{B}_2' = -2(1-t)\vec{x}_0 + 2(1-2t)\vec{x}_1 + 2t\vec{x}_2$

so the tangent direction at $t=0$ is

$\vec{B}_2'(0) = -2\vec{x}_0 + 2\vec{x}_1 \Rightarrow$ tangent line $\vec{y}_0(\lambda) = \vec{x}_0 + 2\lambda(\vec{x}_1 - \vec{x}_0)$

and at $t=1$ is

$\vec{B}_2'(1) = -2\vec{x}_1 + 2\vec{x}_2 \Rightarrow$ tangent line $\vec{y}_1(\mu) = \vec{x}_2 + 2\mu(\vec{x}_2 - \vec{x}_1)$.

Notice that when $\lambda = \frac{1}{2}$ and $\mu = -\frac{1}{2}$ we get $\vec{y}_0 = \vec{y}_1 = \vec{x}_1$, so this is the intersection point.

e.g. $\vec{B}_3(t) = (1-t)^3\vec{x}_0 + 3t(1-t)^2\vec{x}_1 + 3t^2(1-t)\vec{x}_2 + t^3\vec{x}_3$.

Again, $\vec{B}_3(0) = \vec{x}_0$, $\vec{B}_3(1) = \vec{x}_3$.

This time, the tangent to $\vec{B}_3$ at $\vec{x}_0$ passes through $\vec{x}_1$, and the tangent at $\vec{x}_3$ through $\vec{x}_2$. → Problem sheet!

Properties

1). A Bézier curve always lies within the convex hull of the control points x_i, because $\vec{B}_n(t)$ is a weighted average of the $\vec{x}_i$.

(convex hull: the smallest convex set containing all of the points.)

2). If the $\vec{x}_i$ lie on a curve γ, then $\vec{B}_n(t)$ will converge to γ as $n \to \infty$ (cf. last lecture).

3). Bézier curves have a variation diminishing property: If a straight line is drawn through the curve, the number of intersections with the curve will be less than or equal to the number of intersections with the control polygon. (Proof omitted)

Example:- Find the parametric equations of the Bézier curve with control points $(0,0)$, $(0,1)$, $(1,\frac{1}{2})$, $(-1,\frac{1}{2})$.

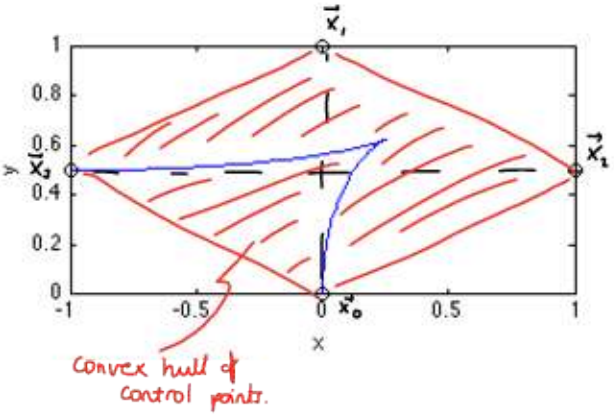
Note: the order matters!

We have

$$x(t) = (1-t)^3(0) + 3t(1-t)^2(0) + 3t^2(1-t)(1) + t^3(-1) = t^2(3-4t)$$

$$y(t) = (1-t)^3(0) + 3t(1-t)^2(1) + 3t^2(1-t)(\tfrac{1}{2}) + t^3(\tfrac{1}{2}) = t\left(3 - 6t + 3t^2 + \tfrac{3}{2}t - \tfrac{3}{2}t^2 + \tfrac{1}{2}t^2\right)$$

$$= t(2t^2 - \tfrac{9}{2}t + 3)$$



Convex hull of control points.

Note that the curve has a cusp where

$x'(t) = y'(t) = 0$

$\Leftrightarrow 3(2t-1)(t-1) = 6t(1-2t) = 0 \Leftrightarrow t = \frac{1}{2}$

which corresponds to

$x(\frac{1}{2}) = \frac{1}{4}$

$y(\frac{1}{2}) = \frac{5}{8}$.

Note: Bézier curves can also have self-intersections, e.g.

You can see the variation diminishing property.

Applications:-

- Bézier curves (and splines) were developed independently by Pierre Bézier (1910-99) of Renault cars and Paul de Casteljau of Citroën.
- They are fundamental to computer aided design and manufacturing.

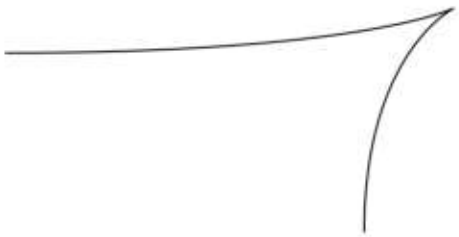
Example:- Bézier curves are used in scalable fonts, e.g. Type 1 or TrueType fonts, and in Adobe Postscript.

→ used because of de Casteljau's efficient algorithm for computing them.

You can easily draw cubic Bézier curves in Postscript, e.g. here is a postscript file for the previous example:

```
%!PS
newpath
400 200 moveto
400 400 600 300 200 300 curveto
stroke
```

($\vec{x}_0$: 400 200; $\vec{x}_1$: 400 400; $\vec{x}_2$: 600 300; $\vec{x}_3$: 200 300)



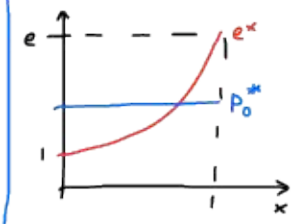
2.3). MINIMAX APPROXIMATIONS

idea goes back to Poncelet (1835) and Chebyshev (1850s).

Aim:- Find a polynomial $p_n \in \mathcal{P}_n$ that minimises the maximum error $\|f - p_n\|_\infty$.

Note: Thm 2.1 tells us that $\|f - p_n\|_\infty$ can be made arbitrarily small for n large enough, but in practice we must work with limited n.

Example:- Find the minimax constant approximation $p_0^* \in \mathcal{P}_0$ to e^x on $[0,1]$.



We want to minimise $\|e^x - p_0\|_\infty$ over all constants p_0. Now

$$\|e^x - p_0\|_\infty = \max_{x\in[0,1]} |e^x - p_0| = \max\{|e - p_0|, |1 - p_0|\}.$$

Notice that the error changes sign so we minimise the maximum error by balancing the error at both ends, ie.

$$e - p_0^* = -(1 - p_0^*) \quad \Rightarrow \quad p_0^* = \frac{e+1}{2}.$$

Clearly p_0^* is unique.

Example:- Find the minimax linear approximation $p_1^* \in \mathcal{P}_1$ to e^x on $[0,1]$.

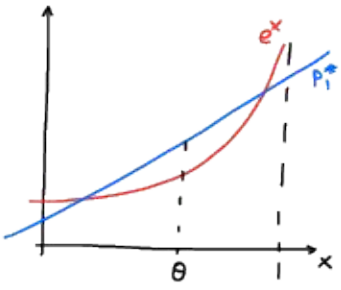
Let $p_1(x) = a + bx$.

We want to minimise $\max_{x\in[0,1]} |e^x - (a+bx)|$.

Local maxima occur at three places,

$x = 0$, $x = \theta$, $x = 1$.

(θ: to be determined)



The maximum error will be minimised if the 3 maxima are equal:

$x = 0 \to \quad e^0 - (a+0) = E \quad (1)$

$x = \theta \to \quad e^\theta - (a + b\theta) = -E \quad (2)$

$x = 1 \to \quad e - (a+b) = E. \quad (3)$

We get a fourth condition from the fact that the error has a turning point at $x = \theta$, so

$$\frac{d}{dx}\left(e^x - (a+bx)\right)\Big|_{x=\theta} = 0 \quad \Rightarrow \quad e^\theta = b. \quad (4)$$

Then

(1) and (3) $\Rightarrow \quad 1 - a = e - a - b \quad \Rightarrow \quad b = e - 1 \simeq 1.718$

$\Rightarrow \quad \theta = \text{Log}\, b.$

(2) and (3) $\Rightarrow \quad e^\theta - a - b\theta = -e + a + b \quad \Rightarrow \quad a = \frac{1}{2}(e - b\theta) \simeq 0.894.$

So the minimax straight line is

$$p_1^*(x) = 0.894 + 1.718x.$$

We will see how to prove that the error in p_n^* has to alternate in sign like this, as well as how to calculate p_n^* more efficiently.

Actually we can already do it when f is a polynomial...

Recall that the Chebyshev polynomials $T_n(x) = \cos(n \arccos(x))$ have the property that

$$\left\| \frac{1}{2^n} T_{n+1} \right\|_\infty \leq \| w_{n+1} \|_\infty \quad \text{for } x \in [-1, 1]$$

↑ kills leading coeff.

where w_{n+1} is any monic degree $n+1$ polynomial (Thm 0.3).

How can we use this? Let $f(x) = x^{n+1} + a_n x^n + \dots + a_0$ be a monic polynomial of degree $n+1$.

Let

$$p_n^*(x) = f(x) - \frac{1}{2^n} T_{n+1}(x).$$

↑ note: x^{n+1} cancels.

Then

$$\| f - p_n^* \|_\infty = \left\| \frac{1}{2^n} T_{n+1} \right\|_\infty \leq \| w_{n+1} \|_\infty$$

for any monic polynomial w_{n+1}.

Since f is monic, this implies that ⟩ $f - w_{n+1} := p_n$

$$\| f - p_n^* \|_\infty \leq \| f - p_n \| \quad \text{for any } p_n \in \mathbb{P}_n.$$

Example:- Find the minimax polynomial $p_2^* \in \mathbb{P}_2$ for $f(x) = x^3 + ax^2 + bx + c$.

We compute $T_3(x)$ from the recurrence:

$T_0(x) = 1$

$T_1(x) = x$

$T_2(x) = 2xT_1(x) - T_0(x) = 2x^2 - 1$

$T_3(x) = 2xT_2(x) - T_1(x) = 4x^3 - 3x$

So the minimax approximation of degree 2 to f is

$$\begin{aligned} p_2^*(x) &= f(x) - \tfrac{1}{2^2} T_3(x) \\ &= x^3 + ax^2 + bx + c - x^3 + \tfrac{3}{4}x \\ &= ax^2 + (b + \tfrac{3}{4})x + c. \end{aligned}$$

This can be extended to any $f \in \mathbb{P}_{n+1}$ ← not just monic. using the fact that $\alpha p_n^*(x)$ is the minimax approximation to $\alpha f(x)$.

2.4). ALTERNATING SETS

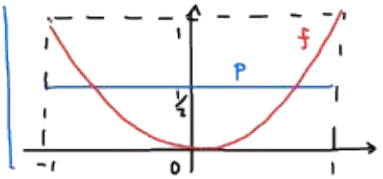
We have seen that the maximum error in the minimax polynomial p_n^* seems to occur $n+2$ times, with alternating sign of $f - p^*$.

Def^n :- Let $f \in C[a,b]$ and p be a polynomial approximation. An *alternating set* / *alternant* of f, p of length n is a sequence of points $x_0, \dots, x_{n-1}$ such that

1). $a \le x_0 < x_1 < \dots < x_{n-1} \le b$.

2). $f(x_i) - p(x_i) = (-1)^i E$ for $i = 0, \dots, n-1$ where either $E = \|f - p\|_\infty$ or $E = -\|f - p\|_\infty$.

Example :- Let $f(x) = x^2$ on $[-1, 1]$ and $p(x) = \frac{1}{2}$.



Here we have $\|f - p\|_\infty = \frac{1}{2}$.
The points $\{-1, 0, 1\}$ form an alternating set of length 3, with
$E = \|f - p\|_\infty = \frac{1}{2}$.
↳ will see that this implies that $p = \frac{1}{2}$ is the minimax degree-1 polynomial.

Before we can prove our general result (that maximum error in minimax polynomial always alternates), we need some preliminary results.

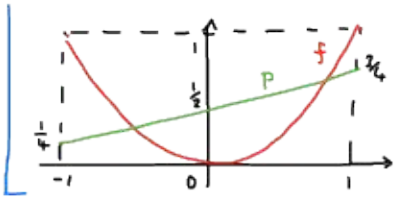
Def^n :- Let $f \in C[a,b]$ and p be a polynomial. A *non-uniform alternating set* of length n is a sequence of points $x_0, \dots, x_{n-1}$ such that

1). $a \le x_0 < x_1 < \dots < x_{n-1} \le b$.

2). $f(x_i) - p(x_i) = (-1)^i e_i$ for $i = 0, \dots, n-1$,

where the e_i's are either all positive or all negative.

Example :- $f(x) = x^2$ on $[-1, 1]$ and $p(x) = \frac{x}{4} + \frac{1}{2}$



Now $\{-1, 0, 1\}$ is a non-uniform alternating set of length 3, with
$e_0 = \frac{3}{4}$, $e_1 = \frac{1}{2}$, $e_2 = \frac{1}{4}$.

The following result lets us bound $\|f - p\|_\infty$ if we can find a suitable non-uniform alternating set.

Thm 2.2 (de la Vallée Poussin) ——— Let $f \in C[a,b]$, $q_n \in P_n$ and p_n^* the minimax polynomial of degree n for f on $[a,b]$. If f, q_n have a non-uniform alternating set of length $n+2$ (critical), then

$$\|f - p_n^*\|_\infty \ge \min_{i \in [0, n+1]} |e_i|.$$

($|e_i| = |f(x_i) - q_n(x_i)|$ where x_i are the n-u.a. set)

- We see that this holds in the above example, where $\min |e_i| = \frac{1}{4}$ and $\|f - p_1^*\|_\infty = \frac{1}{2}$.

↖ De la Vallée Poussin is most famous for proving the Prime Number Thm.

Proof:- (by contradiction).

Assume that $\|f - p_n^*\|_\infty < \min_{i\in[0,n+1]} |e_i|$. ✱

Let $X = \{x_0, \dots, x_{n+1}\}$ be the non-uniform alternating set for f, q_n.

Define

$$r_n(x) = p_n^*(x) - q_n(x) \in P_n.$$

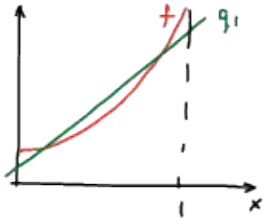
$$= \left(f(x) - q_n(x)\right) - \left(f(x) - p_n^*(x)\right)$$

so $r_n(x_i)$ has the same sign as $f(x_i) - q_n(x_i)$ by ✱.

But then r_n changes sign $n+1$ times, which is a contradiction. ▨

If we don't know p_n^* we can use Thm 2.2 to estimate the error in p^* by choosing a suitable q_n.

Example:- Let $f(x) = e^x$ on $[0,1]$ and $q_1(x) = 0.9 + 1.6x$.



$f - q_1$ changes sign twice in $[0,1]$ and $\{0, 0.5, 1\}$ is a non-uniform alternating set, with

$f(0) - q_1(0) = 0.1$

$f(0.5) - q_1(0.5) = -0.05$

$f(1) - q_1(1) = 0.22$

Hence by Thm 2.2 we have

$$\|f - p_1^*\|_\infty \geq 0.05 \text{ in } [0,1].$$

We can also get an upper bound using the fact that $\|f - p_1^*\|_\infty \leq \|f - q_1\|_\infty$. We need to find $\|f - q_1\|_\infty$:

$f - q_1 = e^x - 0.9 - 1.6x \Rightarrow f' - q_1' = e^x - 1.6$.

Turning point is at $x = \ln(1.6)$.

We have

$$f(0) - q_1(0) = 0.1, \quad f(\ln(1.6)) - q_1(\ln(1.6)) = -0.05, \quad f(1) - q_1(1) = 0.22,$$

so $\|f - q_1\|_\infty = 0.22$.

Hence

$$0.05 \leq \|f - p_1^*\|_\infty \leq 0.22.$$ ✱

We can check this using our result from last lecture,

$$p_1^*(x) = 0.894 + 1.718x.$$

The maximum error was at $x = 0$, so $\|f - p_1^*\|_\infty = |1 - 0.894| = 0.106$ — consistent with ✱.

2.5). CHEBYSHEV EQUIOSCILLATION THEOREM

Thm 2.3 – Let $f \in C[a,b]$. Then $p_n \in \mathbb{P}_n$ is a minimax polynomial of degree n for f if and only if f, p_n have an alternating set of length $n+2$.

not a non-uniform alternating set.

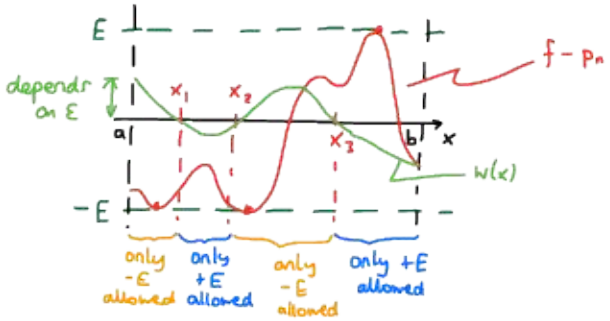
actually was fully proved by Kirchberger (ca. 1902) in his PhD thesis under Hilbert.

Proof:-

(⇐) Suppose that f, p_n have an alternating set of length $n+2$, with
$|f(x_i) - p_n(x_i)| = |E| = \|f - p_n\|_\infty$.
Then De la Vallée Poussin (Thm 2.2) $\Rightarrow \|f - p_n^*\|_\infty \geq |E|$.
But $\|f - p_n^*\|_\infty \leq |E|$ by definition of p_n^*, so $\|f - p_n\|_\infty = \|f - p_n^*\|_\infty$, i.e. p_n is a minimax polynomial.

(⇒) We will assume that f, p_n have no alternating set of length $n+2$, and show that p_n cannot be the best approximation.
Let $E = \|f - p_n\|_\infty$, and assume (w.l.o.g.) that the leftmost extremum has $f - p_n = -E$.
Then there are numbers $x_1, \dots, x_k$ such that $a < x_1 < \dots < x_k < b$ with $k \leq n$, such that
1). $f(x) - p_n(x) < E$ for $x \in [a, x_1] \cup [x_1, x_2] \cup [x_4, x_5] \cup \dots$
2). $f(x) - p_n(x) > -E$ for $x \in [x_1, x_3] \cup [x_3, x_4] \cup \dots$

e.g. here I can find suitable points x_1, x_2, x_3 ($k=3$):



Now let $w(x) = (x_1 - x)(x_2 - x) \cdots (x_k - x)$.
Then
$q_n(x) = p_n(x) + \varepsilon w(x)$ will be a better approximation than p_n for small enough $\varepsilon > 0$ (because all of the extrema will be reduced). Hence p_n cannot be minimax. ∎

you always add something of the correct sign.

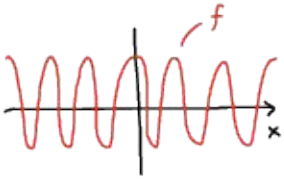
[Note: if there were an alternating set of length $n+2$, we could find $x_1, \dots, x_{n+1}$ so $w(x)$ would have degree $n+1$. Then q_n would not be in $\mathbb{P}_n$ so the argument would fail. ⟶ i.e. we would need a higher degree polynomial to do better.].

Note: The minimax polynomial may have degree $< n$, and may have an alternating set of length $> n+2$.

Example:- $f(x) = \cos(x)$ on $[-n\pi, n\pi]$, find minimax polynomial of degree n.

There is an obvious alternating set of length $2n+1$ when $p_n(x) = 0$.

By Thm 2.3, we must have $p_n^*(x) = 0$.



Corollary 2.4 – The minimax polynomial $p_n^* \in \mathbb{P}_n$ for $f \in C[a,b]$ is unique. (uniqueness)

Proof:- Let both p_n^*, q_n^* be minimax polynomials. Then

$$\left\| f - \frac{p_n^* + q_n^*}{2} \right\|_\infty = \left\| \frac{f - p_n^*}{2} + \frac{f - q_n^*}{2} \right\|_\infty \leq \frac{1}{2}\|f - p_n^*\|_\infty + \frac{1}{2}\|f - q_n^*\|_\infty = \|f - p_n^*\|_\infty.$$

(because both are minimax.)

ie. $\frac{1}{2}(p_n^* + q_n^*)$ is a minimax polynomial.

By Thm 2.3, there are $n+2$ alternating points at which $\frac{1}{2}(f - p_n^*) + \frac{1}{2}(f - q_n^*) = \|f - p_n^*\|_\infty$.

At each of these points, $f - p_n^*$ and $f - q_n^*$ are both $\|f - p_n^*\|_\infty$ or both $-\|f - p_n^*\|_\infty$.

So $f - p_n^*$ and $f - q_n^*$ agree at $n+2$ points, so

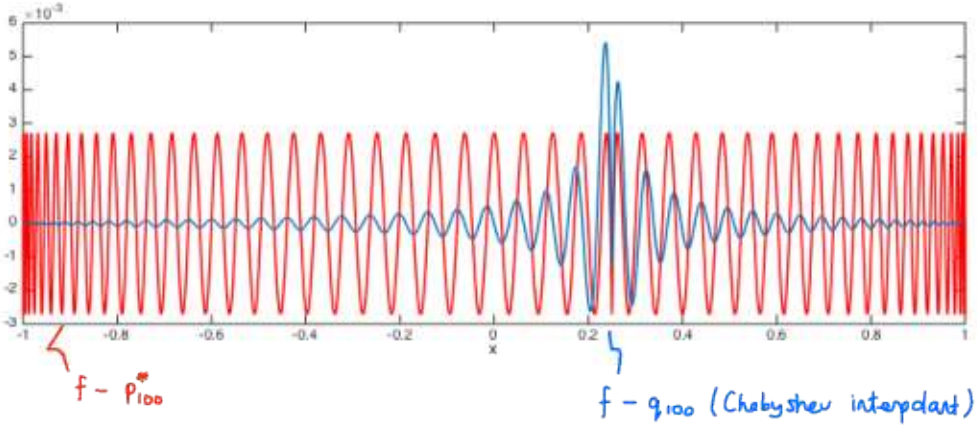
$(f - p_n^*) - (f - q_n^*) = q_n^* - p_n^* = 0$ at these $n+2$ points.

Since $q_n^* - p_n^* \in \mathbb{P}_n$, we must have $q_n^* - p_n^* = 0$. ▨

Remark:- Minimax approximations are not used much in practice, despite being nice in theory.

Note that e.g. Chebyshev interpolants may do better at most locations.

e.g. $f(x) = |x - \frac{1}{4}|$ on $[-1, 1]$.



2.6). REMEZ / EXCHANGE ALGORITHM

Unfortunately the Equioscillation Thm 2.3 doesn't tell us what the minimax polynomial is.

Suppose we have $n=1$, so $p_1^*(x) = a_0 + a_1 x$, and suppose we have an alternating set $\{x_0, x_1, x_2\}$ for f, p_1^*, with $|E| = \|f - p_1^*\|_\infty$. Then the coefficients a_0, a_1 satisfy

$$\begin{aligned} f(x_0) - (a_0 + a_1 x_0) &= E \\ f(x_1) - (a_0 + a_1 x_1) &= -E \\ f(x_2) - (a_0 + a_1 x_2) &= E. \end{aligned}$$

← cf. our initial examples.

i.e.

$$\begin{aligned} a_0 + a_1 x_0 + E &= f(x_0) \\ a_0 + a_1 x_1 - E &= f(x_1) \\ a_0 + a_1 x_2 + E &= f(x_2). \end{aligned}$$

Since x_i, $f(x_i)$ are known, this linear system would give a_0, a_1, E.

↖ If the x_i are distinct, there will be a unique solution.

Unfortunately, we don't usually know an alternating set to start with. The idea of the Remez / Exchange algorithm is iterative:

the size an A. set would need to be.

Step 1). Solve the linear system over a specified reference set of $n+2$ ordered points $\{x_i\}$.

Step 2). Update the reference set by an exchange procedure, and return to Step 1.

To update the reference set, we look for a set $\{y_i\}$, $i = 0, \dots, n+1$, where

1). The existing error $f - p_n$ alternates sign on the y_i.

2). $|f(y_i) - p(y_i)| \geq |E|$ ← found in step 1. at every point y_i. ← the estimate from Step 1.

3). $|f(y_i) - p(y_i)| > |E|$ for at least one y_i.

Example: $n=1$, $f(x) = e^x$ on $[0,1]$. ← previously we found $p_1^*(x) = 0.894 + 1.718x$.

To illustrate the algorithm, start with $x_0 = 0$, $x_1 = \frac{1}{2}$, $x_2 = 1$.

Step 1:-

$$\begin{cases} a_0 + E = e^0 = 1 \\ a_0 + \frac{1}{2}a_1 - E = e^{1/2} \\ a_0 + a_1 + E = e \end{cases} \Rightarrow \begin{aligned} a_0 &= 0.8948 \\ a_1 &= 1.7183 \\ E &= 0.1052 \end{aligned}$$

→ already quite good!

Step 2:- Update the reference set.

A good way to do this is to find the point of maximum $|f - p_1|$ and swap this into the set.

Here,

$$f(x) - p_1(x) = e^x - 0.8948 - 1.7183x$$

$$\Rightarrow \frac{\partial}{\partial x}(f(x) - p_1(x)) = e^x - 1.7183$$ — turning point at $x = \ln(1.7183) = 0.5413$.

We have

$$f(0.5413) - p_1(0.5413) = -0.1067$$ ← close to optimum.

$$f(1) - p_1(1) = 0.1052.$$

So we swap $x_1 = \frac{1}{2}$ for $x_1 = 0.5413$.

Step 1 :- Now solve

$$\begin{cases} a_0 + E = 1 \\ a_0 + 0.5413\,a_1 - E = e^{0.5413} \\ a_0 + a_1 + E = e \end{cases} \Rightarrow \begin{array}{l} a_0 = 0.8941 \\ a_1 = 1.7183 \\ E = 0.1059 \end{array}$$

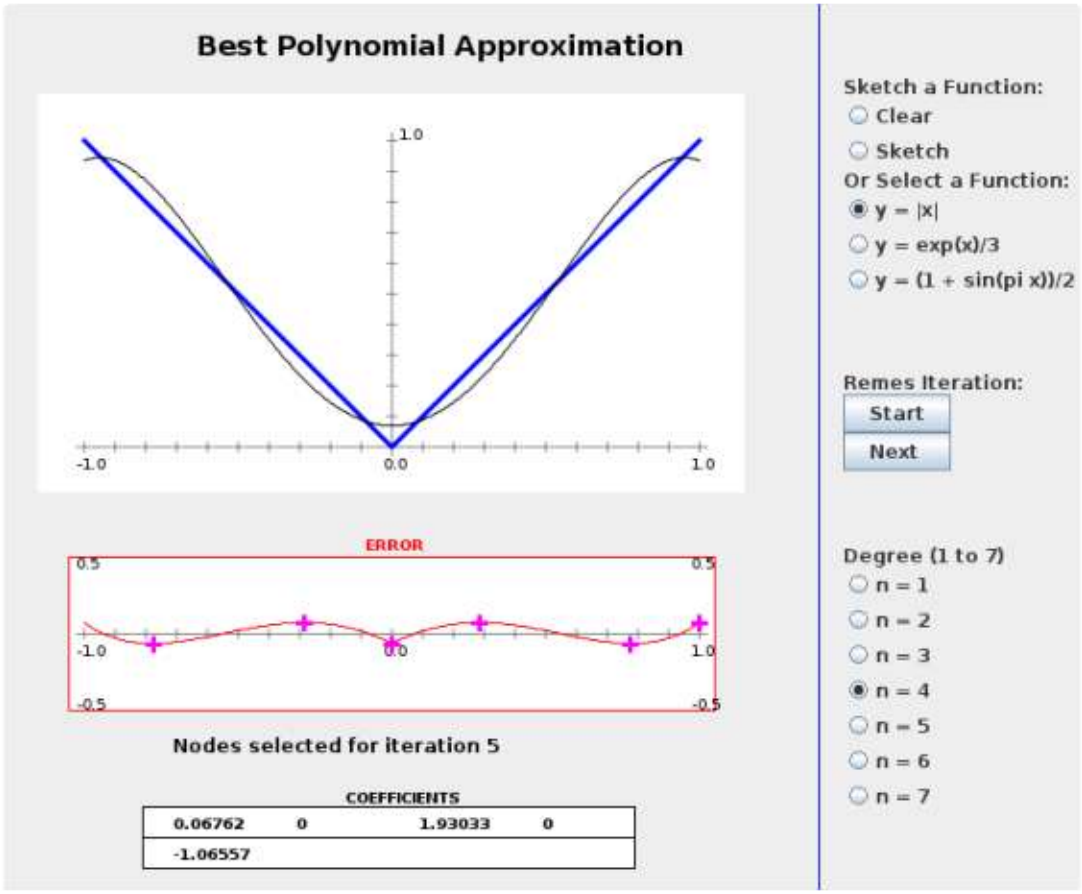
← converging

(note: E has increased — good because we want the alt. set with maximum E !)

In general, the Remez algorithm is found to converge if f is sufficiently smooth and the initial reference set is sufficiently close.

A common way to update the reference set is to find all of the local maxima/minima of $f - p_n$, and take a subset of these that alternate in sign.

e.g. Mayans (2006) — online article in J. Online Math. & Applns. → see app in §9.



3). TRIGONOMETRIC INTERPOLATION

Many real-life functions (e.g. sounds) are periodic, meaning

$$f(x + 2\pi) = f(x) \quad \forall x \in \mathbb{R}.$$ ← by suitably rescaling x we can convert any other period to 2π.

Such functions are well-approximated by trigonometric polynomials

$$p_n(x) = \sum_{k=0}^{n-1} \left(a_k \cos(kx) - b_k \sin(kx) \right),$$

called degree n if a_n or b_n are the highest non-zero coefficients.

Using Euler's identity $e^{ikx} = \cos(kx) + i\sin(kx)$, it is nicer to re-write $p_n(x) = \mathrm{Re}\{q_n(x)\}$ for the complex polynomial

$$q_n(x) = \sum_{k=0}^{n-1} (a_k + ib_k)\left(\cos(kx) + i\sin(kx)\right)$$
$$= \sum_{k=0}^{n-1} (a_k + ib_k) e^{ikx} = \sum_{k=0}^{n-1} c_k e^{ikx}.$$

← $= (\cos x + i \sin x)^k$ → explains why it is a "polynomial"

3.1). THE DISCRETE FOURIER TRANSFORM

Consider the problem of interpolating f at n equally-spaced points $x_j = \frac{2\pi j}{n}$ for $j = 0, \dots, n-1$.
i.e. we want

$$q_n(x_j) = f_j \quad \text{for } j = 0, \dots, n-1.$$

This gives

$$\sum_{k=0}^{n-1} c_k e^{ikx_j} = f_j \iff \sum_{k=0}^{n-1} c_k e^{i\frac{2\pi kj}{n}} = f_j \iff \sum_{k=0}^{n-1} c_k \omega^{jk} = f_j \quad \text{where } \omega = e^{i\frac{2\pi}{n}}.$$

This is a system of n linear equations

$$\underbrace{\begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & & \omega^{n-1} \\ \omega^0 & \omega^2 & \omega^4 & & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega^0 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2} \end{bmatrix}}_{F_n} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{bmatrix}.$$

Inverting would give the interpolation coefficients $\vec{c} = F_n^{-1}\vec{f}$.

The matrix F_n is called the Fourier matrix. ← Note: books differ in whether they include the $\frac{1}{\sqrt{n}}$, or swap F_n and F_n^{-1}.

Oo

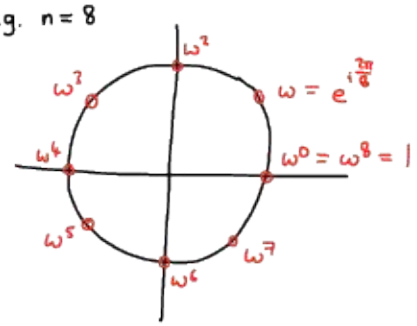
In general, for any n-vector $\vec{x}$, $F_n^{-1}\vec{x}$ is called the discrete Fourier transform (DFT) of $\vec{x}$, and $F_n\vec{x}$ is the inverse DFT of $\vec{x}$. ← the DFT matrix.

The trigonometric interpolation coefficients $\vec{c}$ are given by the DFT of the data F.

Notice that $\omega = e^{i\frac{2\pi}{n}}$ is a primitive n^{th} root of unity, so F_n contains roots of unity.

($\omega^n = 1$ and n is smallest integer of $k = 1, \dots, n$ for which $\omega^k = 1$.)

e.g. $n = 8$



it is "almost" unitary, $F_n^{-1} = F_n^*$ up to $\frac{1}{n}$.

Thm 3.1 — The Fourier matrix is symmetric and satisfies $F_n^{-1} = \frac{1}{n}\overline{F_n}$.

complex conjugate (same as F_n^* since symmetric)

Proof :-

We want to show that $\frac{1}{n}\overline{F_n}F_n = I_n$, i.e.

$$\frac{1}{n}\begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^{-1} & \omega^{-2} & & \omega^{-(n-1)} \\ \omega^0 & \omega^{-2} & \omega^{-4} & & \omega^{-2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega^0 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \cdots & \omega^{-(n-1)^2} \end{bmatrix}\begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & & \omega^{(n-1)} \\ \omega^0 & \omega^2 & \omega^4 & & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega^0 & \omega^{(n-1)} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & 0 & 1 \end{bmatrix}.$$

Let

$$\vec{v}^{(k)} = (1, \omega^k, \omega^{2k}, \dots, \omega^{(n-1)k})$$

denote the k^{th} row of F_n.

Diagonal entries:

$$\frac{1}{n}\left(\overline{F_n}F_n\right)_{kk} = \frac{1}{n}\overline{\vec{v}^{(k)}}\cdot\vec{v}^{(k)} = \frac{1}{n}\sum_{l=0}^{n-1}\overline{\omega^{kl}}\,\omega^{kl} = \frac{1}{n}\sum_{l=0}^{n-1}\omega^{-kl}\omega^{kl} = \frac{1}{n}\sum_{l=0}^{n-1}1 = 1.$$

Off-diagonal entries:

$$\begin{aligned}\frac{1}{n}\left(\overline{F_n}F_n\right)_{jk} = \frac{1}{n}\overline{\vec{v}^{(j)}}\vec{v}^{(k)} &= \frac{1}{n}\sum_{l=0}^{n-1}\overline{\omega^{jl}}\,\omega^{kl} \\ &= \frac{1}{n}\sum_{l=0}^{n-1}\omega^{(k-j)l} \\ &= \frac{1}{n}\sum_{l=0}^{n-1}\left(\omega^{k-j}\right)^l \qquad \text{geometric series.} \\ &= \frac{1}{n}\frac{(\omega^{k-j})^n - 1}{\omega^{k-j} - 1} \\ &= \frac{1}{n}\frac{e^{i(k-j)2\pi} - 1}{e^{i(k-j)\frac{2\pi}{n}} - 1} = 0.\end{aligned}$$

($e^{i(k-j)2\pi} = 1$; $e^{i(k-j)\frac{2\pi}{n}} \neq 1$)

∎

So there is a simple expression for the DFT matrix F_n^{-1}.

Example:- Find the DFT of the vector $\vec{x} = (1, 0, -1, 0)^T$.

Let $\omega = e^{i\frac{2\pi}{4}}$ be a primitive 4^{th} root of unity.

Note that $\omega = e^{i\pi/2} = i$.

The DFT gives

$$F_4^{-1}\vec{x} = \frac{1}{4}\begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^{-1} & \omega^{-2} & \omega^{-3} \\ \omega^0 & \omega^{-2} & \omega^{-4} & \omega^{-6} \\ \omega^0 & \omega^{-3} & \omega^{-6} & \omega^{-9} \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1/2 \\ 0 \\ 1/2 \end{bmatrix}.$$

($\vec{x}$)

Example:- Find a trigonometric polynomial which interpolates the data $(0,1)$, $(\frac{2\pi}{4}, 0)$, $(\frac{4\pi}{4}, -1)$, $(\frac{6\pi}{4}, 0)$.

The complex interpolating polynomial is

$$q_4(x) = \sum_{k=0}^{3} c_k e^{ikx}$$

and the coefficients are given by the DFT

$$\vec{c} = F_4^{-1}\vec{f} = F_4^{-1}\begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1/2 \\ 0 \\ 1/2 \end{bmatrix} \quad \text{from previous example.}$$

Hence

$$q_4(x) = \frac{1}{2}(e^{ix} + e^{3ix}) = \frac{1}{2}(\cos(x) + \cos(3x)) + \frac{1}{2}i(\sin(x) + \sin(3x)).$$

Since the data $\vec{f}$ are real, a real trigonometric polynomial which interpolates the data is

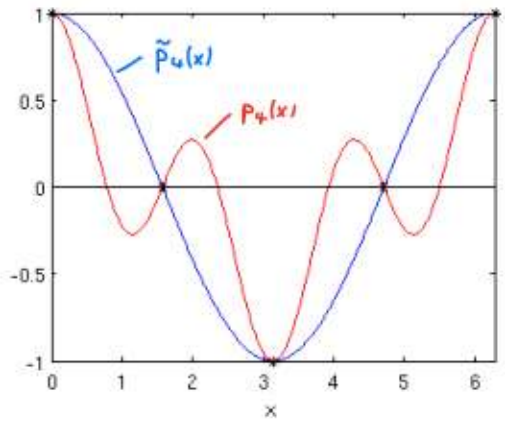
$$p_4(x) = \mathrm{Re}\{q_4(x)\} = \frac{1}{2}(\cos(x) + \cos(3x)).$$

3.2). ALIASING

Trigonometric interpolation as defined so far is not unique.

In the previous example, notice that $\cos(3x_j) = \cos(x_j)$ at all four nodes. So we could also interpolate the data with

$$\tilde{p}_4(x) = \cos(x).$$



This is called aliasing. If there are n nodes, then

$$\cos(k'x_j) = \cos(kx_j) \quad \text{for all } j$$

$$\Leftrightarrow \cos\left(\tfrac{2\pi j}{n}k'\right) = \cos\left(\tfrac{2\pi j}{n}k\right) \quad \text{for all } j$$

$$\Leftrightarrow \frac{k'}{n} = \pm\frac{k}{n} + \mathbb{Z}.$$

(± because $\cos(x) = \cos(-x)$)

e.g. $n=4$: We saw that $\cos(3x_j) = \cos(x_j)$

These are aliases for $n=4$ ($k'=3$, $k=1$) since

$$\frac{3}{4} = -\frac{1}{4} + 1.$$

Other aliases would be e.g. $\cos(5x)$, $\cos(7x)$.

In fact, half of the expansion

$$p_n(x) = \sum_{k=0}^{n-1} \left(a_k \cos(kx) - b_k \sin(kx) \right)$$

is always redundant. For at x_j we have

$$\begin{aligned} \cos\left((n-k)x_j\right) &= \cos\left(2\pi j - \tfrac{2\pi kj}{n}\right) \\ &= \cos(2\pi j)\cos\left(\tfrac{2\pi kj}{n}\right) - \sin(2\pi j)\sin\left(\tfrac{2\pi kj}{n}\right) \\ &= \cos\left(\tfrac{2\pi kj}{n}\right) = \cos(kx_j) \end{aligned}$$

and similarly

$$\begin{aligned} \sin\left((n-k)x_j\right) &= \sin\left(2\pi j - \tfrac{2\pi kj}{n}\right) \\ &= \sin(2\pi j)\cos\left(\tfrac{2\pi kj}{n}\right) - \cos(2\pi j)\sin\left(\tfrac{2\pi kj}{n}\right) \\ &= -\sin(kx_j). \end{aligned}$$

Therefore for n even we can always interpolate the data with the shorter series

$$\tilde{p}_n(x) = a_0 + \sum_{k=1}^{n/2-1} \left((a_k + a_{n-k})\cos(kx) - (b_k - b_{n-k})\sin(kx) \right) + a_{n/2}\cos\left(\tfrac{n}{2}x\right).$$

(and similar for n odd except that there is no "middle" term $k = n/2$).

Example:- In the example above, $a_0 + ib_0 = 0$, $a_1 + ib_1 = \frac{1}{2}$, $a_2 + ib_2 = 0$, $a_3 + ib_3 = \frac{1}{2}$.

so $\tilde{p}_4(x) = (\frac{1}{2} + \frac{1}{2})\cos(x) = \cos(x)$.

This is the idea behind an important result:

not explicitly written down by Nyquist (1920s) but shown by Claude Shannon (1949), the founder of information theory

Thm 3.2 (Nyquist-Shannon Sampling Thm) — A band-limited function $f(x)$ (ie. with no Fourier components of frequency $k > K$) is completely determined by its values $f(x_j)$ for $x_j = \frac{2\pi j}{n}$, $j = 0, \ldots, n-1$, providing that $n > 2K$.

Conversely, for a given n, the highest-frequency fourier component that can be perfectly reconstructed is $k < n/2$. This is called the Nyquist frequency.

Example:- If we had sampled $f(x) = \cos(x)$ with $n > 6$ points, we could be sure that the signal had no $\cos(3x)$ component.

Remark:- If the signal is "sparse" then it may be possible to recover it from fewer samples than required by Thm 3.2 — called compressed sensing.

Some physical examples of aliasing:

1). Moiré patterns in poorly pixelated images (see "Aliasing" in Wikipedia).

2). The "wagon-wheel effect" in video (temporal aliasing).

↳ e.g. if camera shutter clicks 24 times per second and spokes pass vertical 20 times per second, it looks as though wheel is rotating -4 times per second, i.e. backwards!

$$\sin(20x_j) = \sin(-4x_j) \quad \text{for } x_j = \tfrac{2\pi}{24}j, \quad j = 0, \ldots, 23. \qquad \left[\tfrac{20}{24} = -\tfrac{4}{24} + 1\right]$$

3.3). THE FAST FOURIER TRANSFORM

The ubiquity of DFT's in practice is because there is a fast algorithm for computing them.

Applying the DFT means multiplying by F_n^{-1}, which has no non-zero elements so seems to require $O(n^2)$ floating point operations. ← n multiplications for each entry

In fact it is possible to do the transformation in a clever order to reduce the operation count to $O(n \log n)$ — called the Fast Fourier Transform (FFT).

- was known to Gauss, but rediscovered by Cooley & Tukey (1965).
- led to revolution in electronic analysis, control systems and compression.
- in particular, can analyse stream data (on same timescale as data acquired).

The key idea is the following factorisation:

$$F_n = \begin{bmatrix} I_{n/2} & D_{n/2} \\ I_{n/2} & -D_{n/2} \end{bmatrix} \begin{bmatrix} F_{n/2} & 0 \\ 0 & F_{n/2} \end{bmatrix} P_n$$

← important feature: lots of zeros!

where

$$I_{n/2} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}, \qquad D_{n/2} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \omega & & \vdots \\ \vdots & & \omega^2 & \\ 0 & \cdots & & \omega^{n/2-1} \end{bmatrix}$$

ie. a single 1 in each row & column

and $F_{n/2}$ is the Fourier matrix of size $n/2 \times n/2$. The third matrix P_n is a permutation matrix that separates the incoming vector $\vec{c}$ into odd and even parts:

$$P_n \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} c_0 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_1 \\ c_3 \\ \vdots \\ c_{n-1} \end{bmatrix} \begin{matrix} \} \text{ even entries} \\ \\ \\ \\ \} \text{ odd entries} \end{matrix}$$

Example: $n = 4 \quad \Rightarrow \quad \omega = e^{i\pi/2} = i.$

$\omega = e^{i\pi} = -1$

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}, \quad I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D_2 = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The permutation matrix is

$$P_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \rightsquigarrow \quad P_4 \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_2 \\ c_1 \\ c_3 \end{bmatrix}$$

Verify:

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & i \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -i \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & i \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -i \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

$$= F_4.$$

Proof (in general): (assumes n is even).

Let $\omega_n = e^{2\pi i/n}$ (to keep track of which n we are talking about.

The j^{th} entry of $\vec{f} = F_n \vec{c}$ is

$$f_j = \sum_{k=0}^{n-1} \omega_n^{jk} c_k = \sum_{k=0}^{n/2-1} \omega_n^{j2k} c_{2k} + \sum_{k=0}^{n/2-1} \omega^{j(2k+1)} c_{2k+1}$$

(ω_n^{j2k}: even; $\omega^{j(2k+1)}$: odd)

$$= \sum_{k=0}^{n/2-1} (\omega_n^2)^{jk} c_{2k} + (\omega_n)^j \sum_{k=0}^{n/2-1} (\omega_n^2)^{jk} c_{2k+1}$$

$\omega_n^2 = (e^{2\pi i/n})^2 = e^{2\pi i/(n/2)} = \omega_{n/2}$ (*) — Key fact that makes it work.

$$= \sum_{k=0}^{n/2-1} \omega_{n/2}^{jk} c_{2k} + (\omega_n)^j \sum_{k=0}^{n/2-1} (\omega_{n/2})^{jk} c_{2k+1}.$$

- $\sum \omega_{n/2}^{jk} c_{2k}$: apply $F_{n/2}$ to even c_k
- $(\omega_n)^j$: known as a "twiddle factor" (!)
- $\sum (\omega_{n/2})^{jk} c_{2k+1}$: apply $F_{n/2}$ to odd c_k.

Noting that $-\omega_n^{j-n/2} = -\left(e^{\frac{2\pi i}{n}}\right)^{-n/2} \omega_n^j = -e^{-\pi i} \omega_n^j = \omega_n^j$, we see that this gives the matrix factorisation. ▨

The reduction from F_n to two $F_{n/2}$'s cuts the work (almost) in half. But to reduce it much further, we can apply the factorisation recursively, i.e. replace $F_{n/2}$ by two $F_{n/4}$'s, etc.:

$$F_n = \begin{bmatrix} I_{n/2} & D_{n/2} \\ I_{n/2} & -D_{n/2} \end{bmatrix} \begin{bmatrix} F_{n/2} & 0 \\ 0 & F_{n/2} \end{bmatrix} \begin{bmatrix} c_0, c_2, c_4, \dots \\ c_1, c_3, c_5, \dots \end{bmatrix}$$

$$\begin{bmatrix} F_{n/2} & 0 \\ 0 & F_{n/2} \end{bmatrix} = \begin{bmatrix} I_{n/4} & D_{n/4} & 0 & 0 \\ I_{n/4} & -D_{n/4} & 0 & 0 \\ 0 & 0 & I_{n/4} & D_{n/4} \\ 0 & 0 & I_{n/4} & -D_{n/4} \end{bmatrix} \begin{bmatrix} F_{n/4} & 0 & 0 & 0 \\ 0 & F_{n/4} & 0 & 0 \\ 0 & 0 & F_{n/4} & 0 \\ 0 & 0 & 0 & F_{n/4} \end{bmatrix} \begin{bmatrix} c_0, c_4, c_8, \dots \\ c_2, c_6, c_{10}, \dots \\ c_1, c_5, c_9, \dots \\ c_3, c_7, c_{11}, \dots \end{bmatrix}.$$

How many multiplications are used? Suppose $n = 2^m$. Without the FFT, we needed $n^2 = 4^m$ operations. Now we have m stages, from 2^m down to 2^0. Each stage has $n/2$ multiplications to assemble the outputs from diagonal D's. (at 1, no multiplications are needed since $F_1 = 1$).

Hence the operation count is

$$\frac{n}{2} m = \frac{n}{2} \log_2(n).$$

e.g. if $n = 2^{10} = 1024$, then $n^2 = 2^{20} \simeq 1M$,
while $\frac{n}{2} m = 5(1024)$. → major saving for larger matrices!

- The same idea works for the inverse (see Problem Sheet 3).
- A quick way to find the order of the $\vec{c}$'s (after a full recursion) is to write the numbers $0, \dots, n-1$ in base 2 and reverse the order of their bits: (bit-reversal)

e.g. $n = 4$

c_0	00	reverse →	00 c_0
c_1	01		10 c_2
c_2	10		01 c_1
c_3	11		11 c_3.

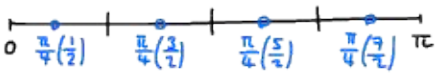
3.4). THE DISCRETE COSINE TRANSFORM

Recall:- When interpolating n points $x_j = \frac{2\pi j}{n}$ in $[0, 2\pi]$ by a function $q_n(x) = \sum_{k=0}^{n-1} c_k e^{ikx}$ we got the DFT $\vec{c} = F_n^{-1}\vec{f}$, where the matrix F_n^{-1} had complex entries.

We can get a matrix of real entries if we interpolate a function on $[0, \pi]$ (no longer needs to be periodic) at n equally-spaced points

$$x_j = \frac{\pi}{n}(j + \tfrac{1}{2}), \quad j = 0, \ldots, n-1$$

e.g. $n = 4$



with a function of the form

$$p_n(x) = \frac{1}{\sqrt{n}} a_0 + \sqrt{\frac{2}{n}} \sum_{k=1}^{n-1} a_k \cos(kx).$$

coefficients are for convenience (to make matrix orthogonal).

This is a real basis.

The interpolation conditions give

$$f_j = p_n(x_j) = \frac{1}{\sqrt{n}} a_0 + \sqrt{\frac{2}{n}} \sum_{k=1}^{n-1} a_k \cos\left(\frac{k\pi}{n}(j + \tfrac{1}{2})\right) \quad \text{for } j = 0, \ldots, n-1,$$

i.e.

$$\vec{f} = C_n \vec{a} \quad \text{where}$$

$$C_n = \sqrt{\frac{2}{n}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \cos\left(\frac{\pi}{n}(\frac{1}{2})\right) & \cos\left(\frac{2\pi}{n}(\frac{1}{2})\right) & \cdots & \cos\left(\frac{(n-1)\pi}{n}(\frac{1}{2})\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{\pi}{n}(\frac{3}{2})\right) & \cos\left(\frac{2\pi}{n}(\frac{3}{2})\right) & & \cos\left(\frac{(n-1)\pi}{n}(\frac{3}{2})\right) \\ \vdots & \vdots & & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{\pi}{n}(\frac{2n-1}{2})\right) & \cdots & & \cos\left(\frac{(n-1)\pi}{n}(\frac{2n-1}{2})\right) \end{bmatrix}$$

($k = 0$, $k = 1$, $k = 2$, $\ldots$, $k = n-1$; $j = 0$, $j = 1$, $\ldots$, $j = n-1$)

To find $\vec{a}$ we need to invert C_n, and the matrix C_n^{-1} is called the discrete cosine transform (DCT).

again there are different definitions / normalisations used.

Thm 3.3 — C_n is orthogonal, ie. $C_n^{-1} = C_n^T$.

Proof:- We can avoid a nightmare of trigonometry with an indirect proof.

Let A_n be the real symmetric circulant matrix

$$A_n = \begin{bmatrix} 1 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ 0 & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}.$$

We will show that the columns $\vec{v}^{(k)}$ of C_n are the eigenvectors of A_n. This means that they are automatically orthogonal, since A_n is real symmetric.

First, $\vec{v}^{(0)} = \frac{1}{\sqrt{2}}(1, 1, \ldots, 1)^T$.

So

$$A_n \vec{v}^{(0)} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ 0 & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \Rightarrow A_n \vec{v}^{(0)} = 0\, \vec{v}^{(0)}$$

(0: eigenvalue)

For every other column, the components take the form

$$v_\ell^{(k)} = \sqrt{\frac{2}{n}} \cos\left(\underbrace{\frac{k\pi}{n}}_{\theta}\left(\ell + \tfrac{1}{2}\right)\right).$$

For $j = 1, \dots, n-2$,

$$\begin{aligned} \left(A_n \vec{v}^{(k)}\right)_j = \sum_{\ell=0}^{n-1} (A_n)_{j\ell}\, v_\ell^{(k)} &= -v_{j-1}^{(k)} + 2v_j^{(k)} - v_{j+1}^{(k)} \\ &= \sqrt{\tfrac{2}{n}}\left(-\cos\left(\theta\left(j-\tfrac{1}{2}\right)\right) + 2\cos\left(\theta\left(j+\tfrac{1}{2}\right)\right) - \cos\left(\theta\left(j+\tfrac{3}{2}\right)\right)\right) \\ &= \sqrt{\tfrac{2}{n}}\left(-\cos(\underbrace{\theta(j+\tfrac{1}{2})}_{\beta} - \theta) + 2\cos(\theta(j+\tfrac{1}{2})) - \cos(\theta(j+\tfrac{1}{2}) + \theta)\right) \\ &= \sqrt{\tfrac{2}{n}}\left(-\cos\beta\cos\theta - \sin\beta\sin\theta + 2\cos\beta - \cos\beta\cos\theta + \sin\beta\sin\theta\right) \\ &= \sqrt{\tfrac{2}{n}}\left(2 - 2\cos\theta\right)\cos\beta. \\ &= \underbrace{\left(2 - 2\cos\left(\tfrac{k\pi}{n}\right)\right)}_{\text{eigenvalues}} v_j^{(k)}. \end{aligned}$$

For $j=0$ we have

$$\begin{aligned} \left(A_n \vec{v}^{(k)}\right)_0 = v_0^{(k)} - v_1^{(k)} &= \sqrt{\tfrac{2}{n}}\left(\cos\left(\tfrac{1}{2}\theta\right) - \cos\left(\tfrac{3}{2}\theta\right)\right) \\ &= \sqrt{\tfrac{2}{n}}\left(\cos\left(\tfrac{1}{2}\theta\right) - \cos\left(\tfrac{1}{2}\theta + \theta\right)\right) \\ &= \sqrt{\tfrac{2}{n}}\left(\cos\left(\tfrac{1}{2}\theta\right) - \cos\left(\tfrac{1}{2}\theta\right)\cos\theta + \underbrace{\sin\left(\tfrac{1}{2}\theta\right)\sin\theta}_{= 2\sin\left(\frac{1}{2}\theta\right)\cos\left(\frac{1}{2}\theta\right)}\right) \\ &= \sqrt{\tfrac{2}{n}}\cos\left(\tfrac{1}{2}\theta\right)\left(1 - \cos\theta + 2\sin^2\left(\tfrac{1}{2}\theta\right)\right) \\ &= \sqrt{\tfrac{2}{n}}\cos\left(\tfrac{1}{2}\theta\right)\left(2 - 2\cos\theta\right) \qquad \swarrow\ 2\sin^2\left(\tfrac{1}{2}\theta\right) = 1 - \cos\theta. \\ &= \left(2 - 2\cos\left(\tfrac{k\pi}{n}\right)\right) v_0^{(k)}. \end{aligned}$$

Similarly,

$$\left(A_n \vec{v}^{(k)}\right)_{n-1} = v_{n-1}^{(k)} - v_{n-2}^{(k)} = \sqrt{\tfrac{2}{n}}\left(\cos\left(\theta\left(n-\tfrac{1}{2}\right)\right) - \cos\left(\theta\left(n-\tfrac{3}{2}\right)\right)\right) = \underbrace{\ldots}_{\text{exercise!}} = \left(2 - 2\cos\left(\tfrac{k\pi}{n}\right)\right) v_{n-1}^{(k)}.$$

To understand where this comes from, note that the columns $\vec{v}^{(k)}$ are discrete approximations of $\cos(kx)$ at the points x_j.

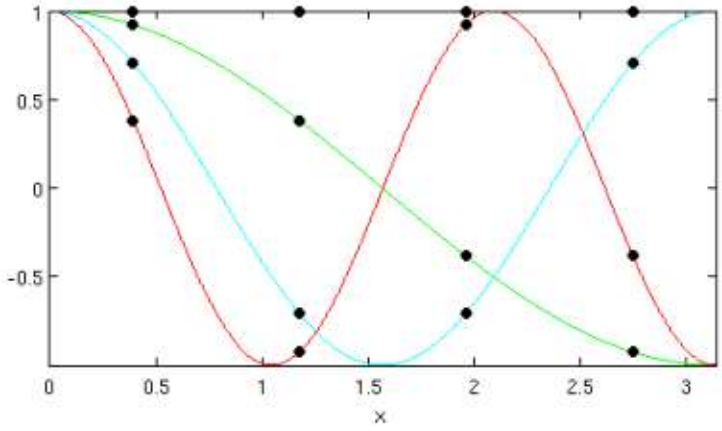
We know that $u(x) = \cos(kx)$ satisfies the differential equation $-\frac{d^2u}{dx^2} = k^2 u$, ie. $\cos(kx)$ is an eigenvalue of the differential operator $-\frac{d^2}{dx^2}$, with eigenvalue k^2.

The matrix A_n is a finite-difference approximation to $\frac{d^2}{dx^2}$ at equally-spaced points,

$$\begin{aligned}(A_n \vec{u})_j &= -u_{j-1} + 2u_j - u_{j+1} \\ &= -\left[(u_{j+1} - u_j) - (u_j - u_{j-1})\right]\end{aligned}$$

so its eigenvectors are discrete cosines.

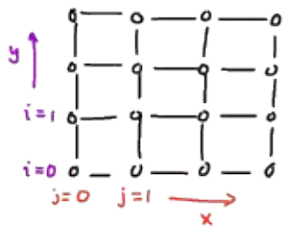
e.g. the columns $\vec{v}^{(0)}, \ldots, \vec{v}^{(3)}$ of C_4 :



3.5) IMAGE COMPRESSION

The DCT lies at the heart of modern compression techniques for images and audio (eg. JPEG, MPEG).

For images, we have a 2-d grid of pixel values (greyscale or colour intensities), so we will need the 2-d DCT. This is simply the 1-d DCT applied in two dimensions, one after the other.



Let $X = (x_{ij})$ be the matrix of pixel values.

- First apply the DCT in the x-direction:

  $C_n^{-1}X^T$ ⟶ resulting columns are DCTs of the rows of X (fixed y_i).

- Now apply a DCT in the y-direction:

$$C_n^{-1}(C_n^{-1}X^T)^T = C_n^{-1}X(C_n^{-1})^T = C_n^{-1}XC_n.$$

ie. the 2-d DCT of a matrix X is $Y = C_n^{-1}XC_n$.

Example:- Find the 2-d DCT of the data

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

The $n=4$ matrix is

$$C_4 = \frac{1}{\sqrt{2}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{2\pi}{8}\right) & \cos\left(\frac{3\pi}{8}\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{3\pi}{8}\right) & \cos\left(\frac{6\pi}{8}\right) & \cos\left(\frac{9\pi}{8}\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{5\pi}{8}\right) & \cos\left(\frac{10\pi}{8}\right) & \cos\left(\frac{15\pi}{8}\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{7\pi}{8}\right) & \cos\left(\frac{14\pi}{8}\right) & \cos\left(\frac{21\pi}{8}\right) \end{bmatrix} = a \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix}$$

where

$$a = \frac{1}{\sqrt{2}}, \quad b = \cos\left(\frac{\pi}{8}\right), \quad c = \cos\left(\frac{3\pi}{8}\right).$$

The 2-d DCT is

$$Y = C_4^{-1}XC_4 = a^2 \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & b & a & -c \end{bmatrix} = \begin{bmatrix} 3 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

It follows that the inverse transform is $X = C_n Y C_n^{-1}$.

Note that (just to make notation nicer)

$$(C_n)_{ij} = \sqrt{\frac{2}{n}}\,\sigma_j \cos\left(\frac{j\pi}{n}(i+\tfrac{1}{2})\right) \quad \text{where} \quad \sigma_j = \begin{cases} 1/\sqrt{2} & \text{if } j=0 \\ 1 & \text{if } j>0. \end{cases}$$

Hence

$$\begin{aligned} x_{ij} &= \sum_{k=0}^{n-1}\sum_{l=0}^{n-1} (C_n)_{ik}\, y_{kl}\, (C_n^{-1})_{lj} \\ &= \sum_{k=0}^{n-1}\sum_{l=0}^{n-1} (C_n)_{ik}\, y_{kl}\, (C_n)_{jl} \\ &= \frac{2}{n}\sum_{k=0}^{n-1}\sum_{l=0}^{n-1} y_{kl}\,\sigma_k\sigma_l \cos\left(\frac{k\pi}{n}(i+\tfrac{1}{2})\right)\cos\left(\frac{l\pi}{n}(j+\tfrac{1}{2})\right). \end{aligned}$$

In other words, the y_{kl} are interpolation coefficients of a 2-d trigonometric polynomial

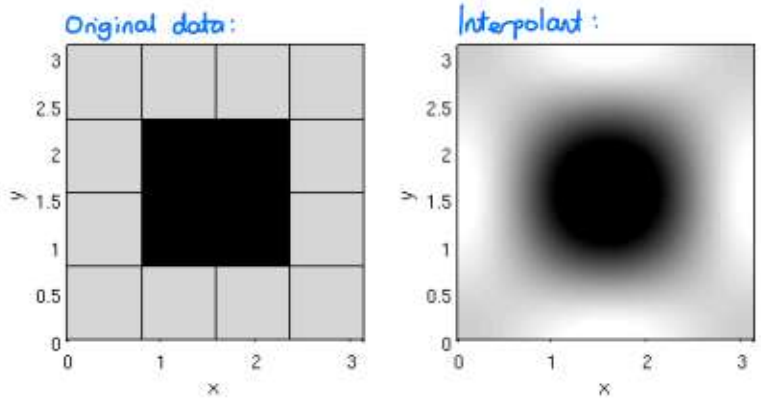
$$p_n(x,y) = \frac{2}{n} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} y_{kl}\, \sigma_k \sigma_l \cos(lx)\cos(ky)$$

for nodes (x_j, y_i) given by $x_j = \frac{\pi}{n}(j+\frac{1}{2})$, $y_i = \frac{\pi}{n}(i+\frac{1}{2})$.

Example :- The interpolation function for the previous example is

$$\begin{aligned} p_4(x,y) = \frac{1}{2}\Big\{ & \frac{1}{2} y_{00} + \frac{1}{\sqrt{2}} y_{01}\cos(x) + \frac{1}{\sqrt{2}} y_{02}\cos(2x) + \frac{1}{\sqrt{2}} y_{03}\cos(3x) \\ & + \frac{1}{\sqrt{2}} y_{10}\cos(y) + y_{11}\cos(x)\cos(y) + y_{12}\cos(2x)\cos(y) + y_{13}\cos(3x)\cos(y) \\ & + \frac{1}{\sqrt{2}} y_{20}\cos(2y) + y_{21}\cos(x)\cos(2y) + y_{22}\cos(2x)\cos(2y) + y_{23}\cos(3x)\cos(2y) \\ & + \frac{1}{\sqrt{2}} y_{30}\cos(3y) + y_{31}\cos(x)\cos(3y) + y_{32}\cos(2x)\cos(3y) + y_{33}\cos(3x)\cos(3y) \Big\}. \end{aligned}$$

$$= \frac{3}{4} + \frac{1}{2\sqrt{2}}\cos(2x) + \frac{1}{2\sqrt{2}}\cos(2y) - \frac{1}{2}\cos(2x)\cos(2y).$$



So far, this process is fully reversible and requires the same amount of storage (an $n \times n$ matrix). The reason the DCT is used is because it tends to concentrate the information in an image into the top-left of the matrix Y. → The human visual system is more sensitive to these low spatial frequencies.

Basic JPEG

- In a grayscale image each pixel is represented by an integer in $[0, 255]$. (2^8, so 8-bit)
  e.g. 64×64 region requires $8(64^2) = 2^{15} = 32$ k bits of data.
- First subtract 128 from each pixel to centre around 0. → makes better use of the DCT
- Apply DCT to each 8×8 block of pixels:

  $$X \to Y = C_8^{-1} X C_8.$$

  (still fully invertible).
- To compress,

  (1) Round Y to nearest integer.

(2) Apply a "low-pass filter", e.g. set $y_{kl} = 0$ for $k+l \geq 8$ → halves storage

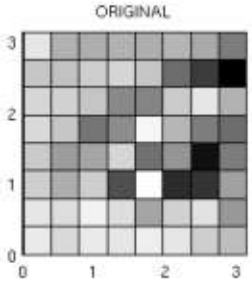
This reduces storage requirements while maintaining qualitative visual aspects of the block.
↳ next time

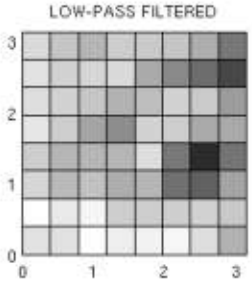
Remarks :-

- A more sophisticated approach is to use "Hilbert qualtization" rather than low-pass filtering.
- Too much compression leads to block artefacts.
- Although sound compression is only 1-d, it is harder because the ear is more sensitive to block artefacts.
  ↳ MPEG uses a modified DCT where blocks can overlap.

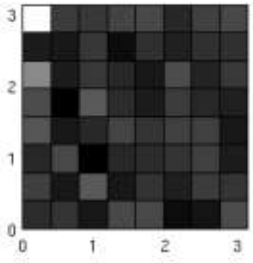
Example :-

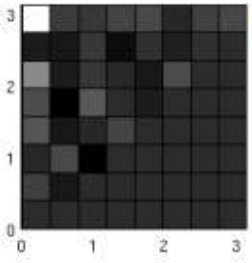
Left eye :





Y





Y after filter

We can see mathematically why the DCT low-pass filter works so well.
For simplicity consider 1-d, so the interpolating trigonometric polynomial is

$$p_n(x) = \sqrt{\frac{2}{n}} \sum_{k=0}^{n-1}{}' a_k \sigma_k \cos(kx), \quad \text{where } \sigma_k = \begin{cases} 1/\sqrt{2} & \text{if } k=0, \\ 1 & \text{if } k>0. \end{cases}$$

Where $\vec{a} = C_n^{-1}\vec{f}$.

Suppose we want to approximate the same data using fewer terms.

Thm. 3.4 (Least-squares property) — Let $\vec{a} = C_n^{-1}\vec{f}$ be the (1-d) interpolation coefficients using the DCT at n points. If $m<n$ then the truncated trigonometric polynomial

$$p_m(x) = \sqrt{\frac{2}{n}} \sum_{k=0}^{m-1}{}' c_k \sigma_k \cos(kx)$$

minimises $\sum_{j=0}^{n-1}{}' (p_m(x_j) - f_j)^2$ (sum of squared error) at the n nodes x_j if $c_0 = a_0, c_1 = a_1, \dots, c_{m-1} = a_{m-1}$.

Recall (2H Numerical Analysis):-

Suppose we want to find $\vec{x}$ to minimise $\|A\vec{x} - \vec{b}\|_2^2$, where A has more rows than $\vec{x}$ has entries.

e.g. $\begin{pmatrix} 3 & 1 \\ 1 & 1 \\ 4 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix}$ → overdetermined so can't solve exactly.

This is equivalent to minimising $(A\vec{x}-\vec{b})^T(A\vec{x}-\vec{b}) = \vec{x}^TA^TA\vec{x} - 2\vec{x}^TA^T\vec{b} + \vec{b}^T\vec{b}$.
This is a non-negative quadratic function of $\vec{x}$, so a minimum exists and is found by setting the partial derivatives $\frac{\partial}{\partial x_i}$ to zero, giving

$$A^TA\,\vec{x} = A^T\vec{b} \quad \text{— normal equations.}$$

If A is orthogonal, then $A^T = A^{-1}$ so these reduce to $\vec{x} = A^T\vec{b}$.

Proof of Thm 3.4:-

We are trying to solve $C_n\vec{c} = \vec{f}$ with C_n orthogonal and $\vec{c} \in \mathbb{R}^m$, so the least-squares solution is simply

$$c_j = \left(C_n^{-1}\vec{f}\right)_j = a_j \quad \text{for } j = 0, \dots, m-1.$$

∎

Example:- Interpolate the data $(\frac{\pi}{8}, 1)$, $(\frac{3\pi}{8}, 0)$, $(\frac{5\pi}{8}, -1)$, $(\frac{7\pi}{8}, 0)$ with the DCT.

We have

$$\vec{a} = C_n^{-1}\vec{f} = \sqrt{\frac{1}{2}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos(\frac{\pi}{8}) & \cos(\frac{3\pi}{8}) & \cos(\frac{5\pi}{8}) & \cos(\frac{7\pi}{8}) \\ \cos(\frac{2\pi}{8}) & \cos(\frac{6\pi}{8}) & \cos(\frac{10\pi}{8}) & \cos(\frac{14\pi}{8}) \\ \cos(\frac{3\pi}{8}) & \cos(\frac{9\pi}{8}) & \cos(\frac{15\pi}{8}) & \cos(\frac{21\pi}{8}) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix} \simeq \begin{bmatrix} 0 \\ 0.9239 \\ 1 \\ -0.3827 \end{bmatrix}$$

So the interpolant is

$$p_4(x) = \frac{1}{2}a_0 + \frac{1}{\sqrt{2}}a_1\cos(x) + \frac{1}{\sqrt{2}}a_2\cos(2x) + \frac{1}{\sqrt{2}}a_3\cos(3x)$$
$$= \frac{1}{\sqrt{2}}\left(0.9239\cos(x) + \cos(2x) - 0.3827\cos(3x)\right).$$

Thm 3.4 implies that the least-squares approximation using the basis $1, \cos(x), \cos(2x)$ is

$$p_2(x) = \frac{1}{\sqrt{2}}\left(0.9239\cos(x) + \cos(2x)\right).$$

Also the least-squares approximation using only $1, \cos(x)$ is

$$p_2(x) = \frac{1}{\sqrt{2}}(0.9239)\cos(x)$$

and using only a constant is

$$p_1(x) = 0.$$