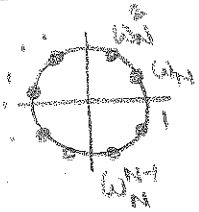
MATH 312
Lecture 28

THE FAST FOURIER TRANSFORM

Recap: The N-th **complex root of unity** is $\omega_N = e^{2\pi i/N}$ or $\cos(2\pi i/N) + i\sin(2\pi i/N)$; and its powers $\omega_N^0, \omega_N^1, \ldots, \omega_N^{N-1}$ generate ALL such roots.



Each of them satisfies $\omega^N = 1$

- Two possible generalizations of linear algebra to ∞ dimensions

1. $\ell_2(\mathbb{C}) = \{$ all complex sequences $(a_0, a_1, a_2, \ldots)$ so that "length" is finite, i.e. $(a_0^2 + a_1^2 + \cdots)^{1/2} < \infty \}$

. . . . . .

2. $L_2[0,2\pi]$ $\{$ all functions $f(t): [0,2\pi] \to \mathbb{C}$ so that "length" is finite, i.e. $\left[\int_0^{2\pi} f(t)\overline{f(t)}\,dt\right]^{1/2} < \infty \}$

FACT • the functions $\{ e^{i\cdot 0\cdot t}, e^{it}, e^{i2t}, e^{i3t}, \ldots \}$ form an ORTHOGONAL BASIS for $L_2[0,2\pi]$. That is, each $f: [0,2\pi] \to \mathbb{C}$ in L_2 is an (infinite!) linear combination:

$$f(t) = a_0 e^{i0t} + a_1 e^{it} + a_2 e^{i2t} + \cdots$$

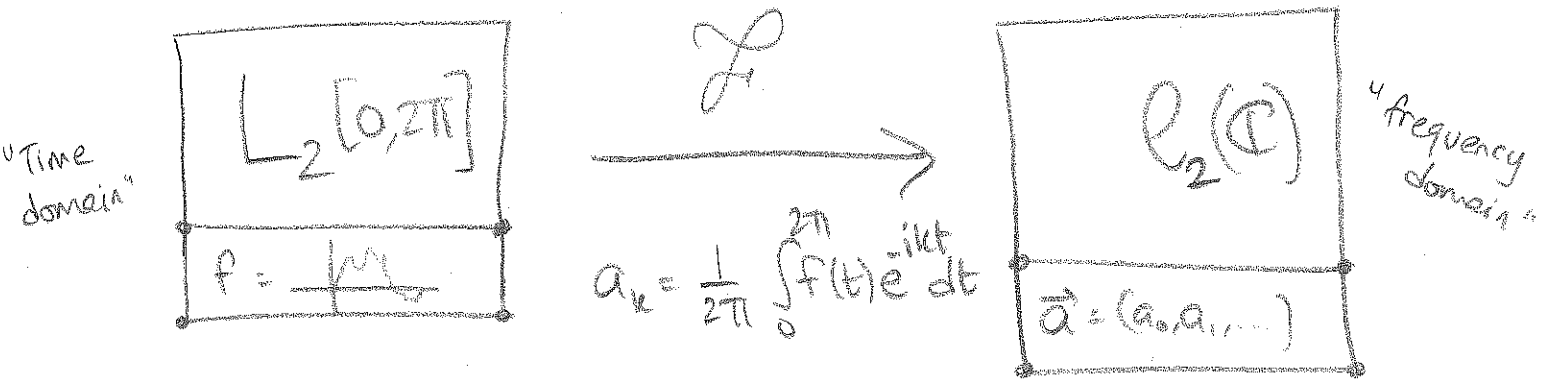
- These coefficients $(a_0, a_1, \ldots)$ are called the **Fourier COEFFICIENTS** of f, and they lie in $\ell_2(\mathbb{C})$!!

Given f, we can compute

$$a_k = \frac{1}{2\pi}\int_0^{2\pi} f(t)\, e^{-ikt}\, dt$$

from ORTHOGONALITY !! $\int_0^{2\pi} e^{imt} e^{-int} = \begin{cases} 2\pi \\ 0 \end{cases}$

This assignment



is called the FOURIER TRANSFORM.

The "inverse" operation $\mathcal{F}^{-1}$ is easy! Starting from $\vec{a} = (a_0, a_1, \dots)$, just set $\boxed{f(t) = \sum_{n=0}^{\infty} a_n e^{int}}$

TODAY: DISCRETE VERSION of $\mathcal{F}$ & $\mathcal{F}^{-1}$

In "practical" situations, we don't have access to the input function $f(t)$ in its entirety: most modern music, for instance, is DIGITAL, as are images. To get a computer to process even the whole interval $[0,2\pi]$ is impossible.

So we chop it up! Let's say into "N" equal pieces

And now, we are only allowed to access the values of $f(t)$ when t is one of these discrete points. For convenience, let's write:

$$t_n = \frac{2\pi}{N} n \quad , \quad n = \{0, \ldots, N-1\}$$

(ignore N: $f(0) = f(2\pi)$)

and: $f_n = f(t_n)$

This has transformed our input from $L_2[0, 2\pi]$ to something <u>much</u> nicer:

$$(f_0, f_1, f_2, \ldots, f_{N-1}) \leftarrow \text{a vector in } \mathbb{C}^N !!$$

But what about the transform itself? We <u>used</u> to have, in the non-discrete case, an integral:

$$a_k = \frac{1}{2\pi} \int_0^{2\pi} f(t) e^{-ikt} \, dt$$

But now we lack the data of $f(t)$ to compute it completely. The best we can do, is to just discretize this integral back into a <u>sum</u>; to separate out the notation from the $\vec{a}$'s, I'll use big-F to denote the coefficients:

$$F_k = \sum_{n=0}^{N-1} f(t_n) e^{-ikt_n}$$

Which, if we expand out $t_n = 2\pi n/N$ becomes this:

$f(t_n)$ ——— t_n

$$F_k = \sum_{n=0}^{N-1} f_n e^{-i\left(\frac{2\pi}{N}\right)nk}$$

Def

This is the DISCRETE FOURIER TRANSFORM. k goes from 0 to N-1, so the output $(F_0, \ldots, F_{N-1})$ is ALSO in $\mathbb{C}^n$.

There's more!

- The exponential stuff: $e^{-i\frac{2\pi}{N}nk}$ should not be mysterious! It is just ω_N^{-nk} where $\omega_N = e^{2\pi i/N}$ is the primitive n-th root of unity... so, $\omega_2 = -1$ and $\omega_4 = i$, etc.
- Of course, there is a DISCRETE FOURIER MATRIX!

$$\begin{bmatrix} F_0 \\ \vdots \\ F_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & & & \\ \vdots & & \omega_N^{-nk} & \\ 1 & & & \end{bmatrix} \begin{bmatrix} f_0 \\ \vdots \\ f_{N-1} \end{bmatrix}$$

(k: columns; n: rows)

Warning!

Both k and n go from 0 to N-1, NOT 1 to N

$N \times N$ matrix, $\mathcal{D}_N$ ("DISCRETE FOURIER MATRIX")

Eg. • When $N=2$, $\omega_2 = -1$

$$\mathcal{D}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} (-1)^{00} & (-1)^{01} \\ (-1)^{10} & (-1)^{11} \end{bmatrix}$$

• When $N=4$, $\omega_4 = i$, so

$$\mathcal{D}_4 = \begin{array}{c} n \downarrow \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{bmatrix} \overset{0}{1} & \overset{1}{1} & \overset{2}{1} & \overset{3}{1} \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \rightarrow k$$

Since we start at 0, This is in row 2, column 1. So, the entry $(\mathcal{D}_N)^{-kn}$ is just $(i)^{-2\cdot 1} = i^{-2} = 1/i^2 = 1/-1 = -1$

PROPERTIES of $\mathcal{D}_N$

1. Symmetric: $\mathcal{D}_N^T = \mathcal{D}_N$ (this is obvious, $\omega_N^{-kn} = \omega_N^{-nk}$)
2. Orthogonal columns (this takes some work)
3. Inverse: $\mathcal{D}_N^{-1} = \frac{1}{N}\overline{\mathcal{D}_N}$ (Mostly a consequence of orthog. columns)

!!

Be careful when checking for ORTHOGONALITY of columns in the case of complex entries. In $\mathcal{D}_4$, columns 1 and 3 appear to NOT be orthogonal:

$$\begin{bmatrix} 1 & -i & -1 & i \end{bmatrix} \begin{bmatrix} 1 \\ i \\ -1 \\ -i \end{bmatrix} = 1 + 1 + 1 + 1 = 4 \neq 0$$

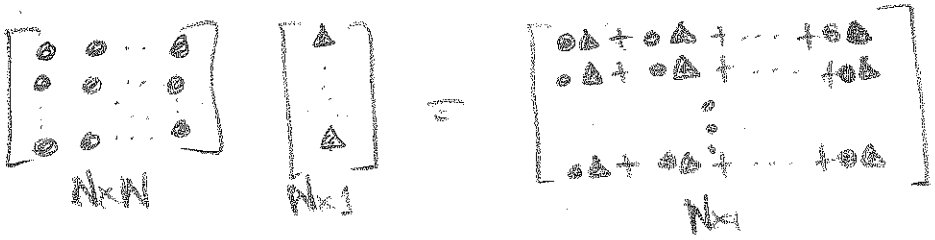
In Complex-land, the correct quantity to test involves conjugating the second vector:

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

$$= 1 + (-1) + 1 + (-1) = 0$$

THE FAST FOURIER TRANSFORM

- Usually, computing a matrix product with vectors, eg,



Requires N^2 multiplications of the form ● ▲

- But, AMAZINGLY, one can multiply by $\mathcal{D}_N$ with only $N \log_2 N$ multiplications! This exploits the special structure of $\mathcal{D}_N$ and does NOT work in general for other matrix-vector products.

- By the way, if $N = 2^{10} = 1024$, then:

  ~ N^2 operations $\geq$ 1 million ops
  ~ $N \log_2 N$ ops $=$ 10240 ops

  } So, $N \log_2 N$ is already a 100 times faster!!

- How does this work??

COOLEY - TUKEY ALGORITHM for FFT.

The basic idea is quite simple: use a matrix factorization. For simplicity, we can assume that N is a power of 2, i.e., $N = 2^k$ for some k. (otherwise, just pad everything with zeros...)

Here's the factorization, sort of:

$$\mathcal{D}_N = \begin{bmatrix} \text{something} \\ \text{"easy"} \\ A \end{bmatrix} \begin{bmatrix} \mathcal{D}_{N/2} & 0 \\ 0 & \mathcal{D}_{N/2} \end{bmatrix} \begin{bmatrix} \text{"something} \\ \text{easy"} \\ B \end{bmatrix}$$

So, the cost of multiplying by $\mathcal{D}_N$ is essentially the cost of two multiplications by $\mathcal{D}_{N/2}$, each of which is another two by $\mathcal{D}_{N/4}$ etc.

So,

$$\boxed{\text{cost of } \mathcal{D}_N \text{ multiplication}} \approx 2 \boxed{\begin{array}{c}\text{Cost of} \\ \mathcal{D}_{N/2} \\ \text{multiplications}\end{array}} \approx 4 \boxed{\begin{array}{c}\text{Cost of} \\ \mathcal{D}_{N/4} \\ \text{multiplications}\end{array}} \quad \text{etc.}$$

The $\approx$ indicates that even the "easy" matrices A and B have a cost to them. Anyway, the formula above gives us

$$\left(\begin{array}{c}\text{Cost of } \mathcal{D}_N \\ \text{multiplication}\end{array}\right) \approx 2^k \left(\begin{array}{c}\text{cost of } \mathcal{D}_{N/2^k} \\ \text{multiplication}\end{array}\right)$$

which is where the "$\log_2 N$" factor comes from. To see what the "EASY" matrices must be, we start with a SIMPLE CASE

WHEN $N = 4$, ...

$$\mathcal{D}_N = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \quad \text{and} \quad \mathcal{D}_{N/2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

So, $\mathcal{D}_{N/2} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x+y \\ x-y \end{bmatrix}$

But $\mathcal{D}_N \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} a+b+c+d \\ a-ib-c+id \\ a-b+c-d \\ a+ib-c-id \end{bmatrix}$

So far, this looks pretty routine... the FANTASTIC part is coming now: group "a" with "c" and "b" with "d" — so,

$$\mathcal{D}_N \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} (a+c) + (b+d) \\ (a-c) - i(b-d) \\ (a+c) - (b+d) \\ (a-c) + i(b-d) \end{bmatrix}$$

The $\begin{bmatrix} (a+c) \\ (a-c) \end{bmatrix}$ parts are JUST $\mathcal{D}_{N/2} \begin{bmatrix} a \\ c \end{bmatrix}$!

And what about the others?

$\begin{bmatrix} b+d \\ -i(b-d) \end{bmatrix}$ is ALMOST $\mathcal{D}_{N/2} \begin{bmatrix} b \\ d \end{bmatrix}$, we just scaled (by the diagonal matrix $\begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$).

Here's the POINT: We are computing $\mathcal{D}_N \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$

The output of $\mathcal{D}_N$ is very similar to that of "Two copies of $\mathcal{D}_{N/2}$", one acting on $\begin{bmatrix} a \\ c \end{bmatrix}$ and another on $\begin{bmatrix} b \\ d \end{bmatrix}$. In fact,

Note ordering!

$$\mathcal{D}_4 \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} Id & T \\ Id & -T \end{bmatrix} \begin{bmatrix} \mathcal{D}_2 & 0 \\ 0 & \mathcal{D}_2 \end{bmatrix} \begin{bmatrix} a \\ c \\ b \\ d \end{bmatrix}$$

Where $T = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix} = \begin{bmatrix} \omega_N^0 & 0 \\ 0 & \omega_N^1 \end{bmatrix}$ is always DIAGONAL, containing increasing negative powers of the primitive root $\omega_4 = i$.

So, the "EASY MATRIX" $A = \begin{bmatrix} Id & T \\ Id & -T \end{bmatrix}$ and the "EASY MATRIX" B just does this:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \longmapsto \begin{bmatrix} a \\ c \\ b \\ d \end{bmatrix}$$ so a PERMUTATION!

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ Flip!

AT LAST $\mathcal{D}_4 = \left[\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -i \\ \hline 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & i \end{array}\right] \left[\begin{array}{cc|cc} \mathcal{D}_2 & & 0 & \\ & & & \\ \hline 0 & & \mathcal{D}_2 & \end{array}\right] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

A · B

And (MUCH) More generally,

$$\mathcal{D}_N = A_N \cdot \begin{bmatrix} \mathcal{D}_{N/2} & 0 \\ 0 & \mathcal{D}_{N/2} \end{bmatrix} B_N ,$$

where $$A_N = \left[\begin{array}{c|c} Id_{N/2 \times N/2} & T_N \\ \hline Id_{N/2 \times N/2} & -T_N \end{array}\right] ,$$

$$\left(T_N = \begin{bmatrix} 1 & & 0 \\ & \omega_N^1 & \\ & & \ddots \\ 0 & & \omega_N^{(N-1)} \end{bmatrix} \text{ is diagonal} \right)$$

and

$$B_N = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{array}\right] \begin{matrix} \} N/2 \\ \} N/2 . \end{matrix}$$

(The matrix which takes in $\begin{pmatrix} x_0 \\ \vdots \\ x_{N-1} \end{pmatrix}$ and spits out $\left(\begin{array}{c} x_{evens} \\ \hline x_{odds} \end{array}\right)$