LECTURE #1.

① Discrete Fourier transform (DFT)

$\mathbb{C}$ adjoint!

- begin with vector $x \in \mathbb{C}^n$, length n, dot prod $\langle x, y\rangle = x^* y$
- let $\mathcal{F}_{j,k}$ be DFT matrix : $j,k = 0, 1, \ldots, n-1$

$$\mathcal{F}_{j,k} = \frac{1}{\sqrt{n}} e^{2\pi i jk/n} = \frac{1}{\sqrt{n}} \left(e^{2\pi i j/n}\right)^k$$

• so $\mathcal{F}$ is a vandermonde matrix; it's invertible.

• also $e^{2\pi i y/n}$ is periodic with period n, so $j,k \in \mathbb{Z}_n$.

Note: this means that we must treat x as a periodic function; funct^n defined on $\mathbb{Z}_n$.

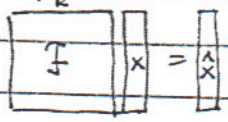
different BCs yield/necessitate different transforms

Discuss: what are some of those transforms?

Discuss: if we have IR valued vectors?

- define $\hat{x} = \mathcal{F} x$ discrete Fourier transform

$$\hat{x}(j) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} e^{2\pi i jk/n} x(k)$$



and $\hat{x}(j+n) = \hat{x}(j)$ so $\hat{x}$ is periodic, too. — $\hat{x}$ is called FT, $\hat{x}(j)$ are Fcoeffs. $\hat{x} \in \mathbb{C}^n$ too.

- To facilitate computations and our derivation/verification of properties of the DFT, here's a useful trigonometric sum:

$$① \quad \sum_{k=0}^{n-1} e^{ikx} = \sum_{k=0}^{n-1} \left(e^{ix}\right)^k = \frac{1-\left(e^{ix}\right)^n}{1-e^{ix}}$$

$$= \frac{1-e^{inx}}{1-e^{ix}} = \left[\frac{\sin\left(\frac{nx}{2}\right)}{\sin\left(\frac{x}{2}\right)} e^{ix(n-1)/2}\right]$$

r is a n^{th} root of unity if $r^n = 1$; it's primitive if n is the smallest int. of $k = 1, 2, \ldots, n$ s.t. $r^k = 1$.

• if $e^{ix} = e^{2\pi i j/n}$ for $e^{2\pi i/n}$ a primitive root of unity,

you'll easily verify that $\frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} e^{2\pi i jk/n} = 0$ for any $j \in \mathbb{Z}_n$.

as $\sin\left(\frac{n}{2} \cdot \frac{2\pi j}{n}\right) = \sin(\pi j) = 0$

let's check:

$$e^{ix} = \cos x + i \sin x$$

$$\Rightarrow \sin x = \frac{1}{2i}\left(e^{ix} - e^{-ix}\right) \Rightarrow \sin\left(\frac{x}{2}\right) = \frac{1}{2i}\left(e^{ix/2} - e^{-ix/2}\right)$$

$$1 - e^{ix} = e^{ix/2}\left(e^{-ix/2} - e^{ix/2}\right) = -e^{ix/2}\left(e^{ix/2} - e^{-ix/2}\right)$$

$$1 - e^{inx} = e^{inx/2}\left(e^{-inx/2} - e^{inx/2}\right) = -e^{inx/2}\left(e^{inx/2} - e^{-inx/2}\right)$$

$$\Rightarrow \frac{1-e^{inx}}{1-e^{ix}} = \frac{+e^{inx/2}\left(e^{inx/2} - e^{-inx/2}\right) \cdot 2i}{+e^{ix/2}\left(e^{ix/2} - e^{-ix/2}\right) \cdot 2i} = \frac{\sin\left(\frac{nx}{2}\right)}{\sin\left(\frac{x}{2}\right)} \cdot e^{ix(n-1)/2} \quad ✓$$

- Let's look at properties of the DFT

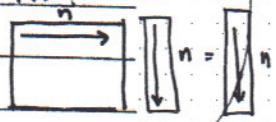
- $\mathcal{F}$ is a unitary matrix (you check that dot prods between cols are zero and cols have unit ℓ_2 norm)

you verify that $\mathcal{F}^{-1} = \mathcal{F}^*$ ~ note: adjoint, $\mathbb{C}$-valued!

- to compute the DFT, you could just do the naive MVM

$\hat{x} = \mathcal{F}x$

which for x of length n, takes $O(N^2)$ flops.



① it's easy to verify that cols have unit ℓ_2 norm

$$\frac{1}{n}\sum_{k=0}^{n-1} \left| e^{2\pi i jk/n} \right|^2 = \frac{1}{n}\sum_{k=0}^{n-1} 1 \cdot \frac{n}{n} = 1.$$

② look at dot prods: $j \neq l$

$$\frac{1}{n}\sum_{k=0}^{n-1} e^{2\pi i jk/n} \, e^{-2\pi i lk/n} = \frac{1}{n}\sum_{k=0}^{n-1} e^{2\pi i k(j-l)/n} = \frac{1}{n}\sum_{k=0}^{n-1} e^{2\pi i km/n} \quad \text{for } m \neq 0$$

$= 0$ (as we already computed)

▶ Discuss: what are some other ONBs for $\mathbb{C}^n$ (or $\mathbb{R}^n$)

- Fast Fourier Transform: $\mathcal{F}$ is a very special matrix; it's got a recursive factorization, structure depends on the factorization of n.

- look at basic case $n = 2^\ell$ (some power of 2), decompose FT into even and odd parts:

$$\hat{x}(j) = \sum_{k=0}^{n/2-1} x(2k) e^{2\pi i j 2k/n} + \sum_{k=0}^{n/2-1} x(2k+1) e^{2\pi i j(2k+1)/n}$$

(even) (odd)

$$= e^{2\pi i j 2k/n} \cdot e^{2\pi i j/n}$$

$$= \sum_{k=0}^{n/2-1} x(2k)\, e^{2\pi i j(2k)/n} + e^{2\pi i j/n} \sum_{k=0}^{n/2-1} x(2k+1) e^{2\pi i j(2k)/n}$$

$$= \hat{x}_{\text{even}}(j) + e^{2\pi i j/n}\, \hat{x}_{\text{odd}}(j) \qquad \text{for } j < n/2$$

$\hat{x}_{\text{even}}(j)$: DFT of size $n/2$ (and $n/2$ periodic) for even elts of x

$\hat{x}_{\text{odd}}(j)$: DFT of size $n/2$ (and $n/2$ periodic) for odd elts of x.

→ note that for $j \geq n/2$

leads to butterfly algo. with twiddle factors on edges...

$$\hat{X}_e(j) = \hat{X}_e(j - n/2)$$

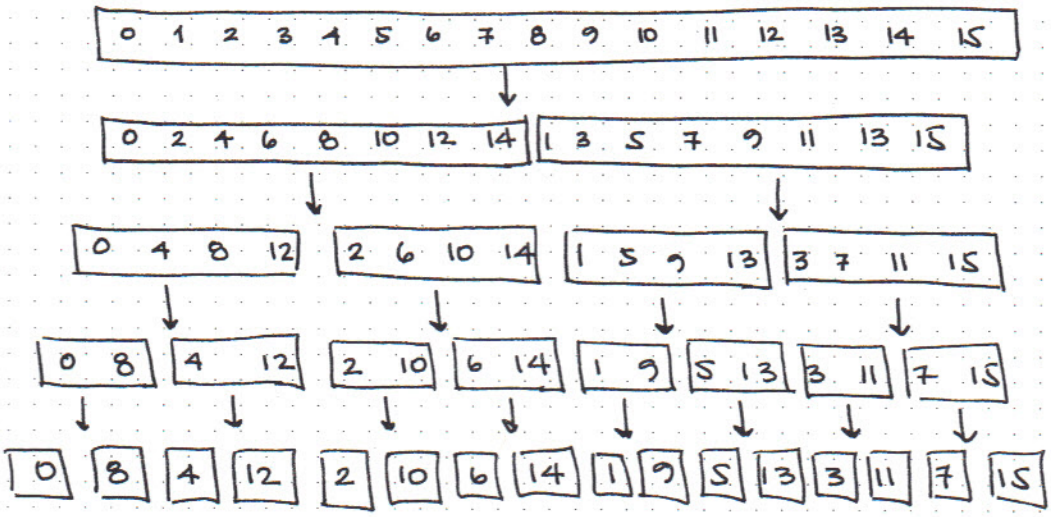
$$\hat{X}_o(j) = \hat{X}_o(j - n/2)$$

and twiddle factor = $e^{\pi i (j - n/2)/n}$

Let's look at how to organize this computation – this is called the butterfly algo. It's recursive: divide signal into 2 halves & recurse

1 signal, N pts. ⟶ N signals, 1 pt.

decimation in time version



↳ how do we arrange these elts? Look at binary expansion of index

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

then rearrange the elts ⟶ according to decomp^n above.

Decimal	Binary
0	0000
8	1000
4	0100
12	1100
2	0010
10	1010
6	0110
14	1110
1	0001
9	1001
5	0101
13	1101
3	0011
11	1011
7	0111
15	1111

STEP 1: reorder samples in bit reversal arrangement

$1 \mapsto 8$

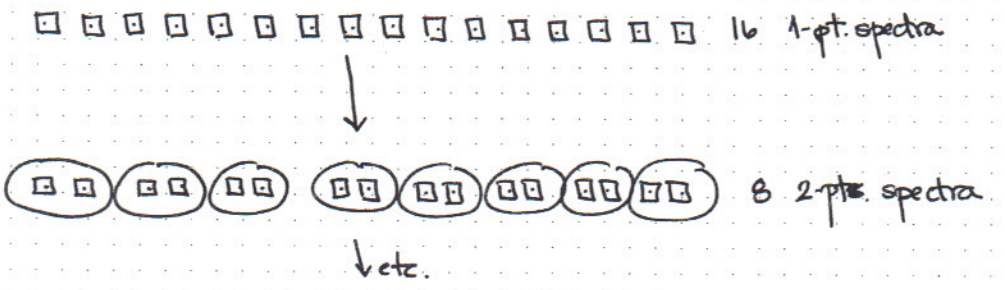
$0001 \mapsto 1000$

STEP 2: find FCs of each of the 1pt. signals. This is trivial! Except you need
↳ think of these vals as being from spectra NOT time domain samples.

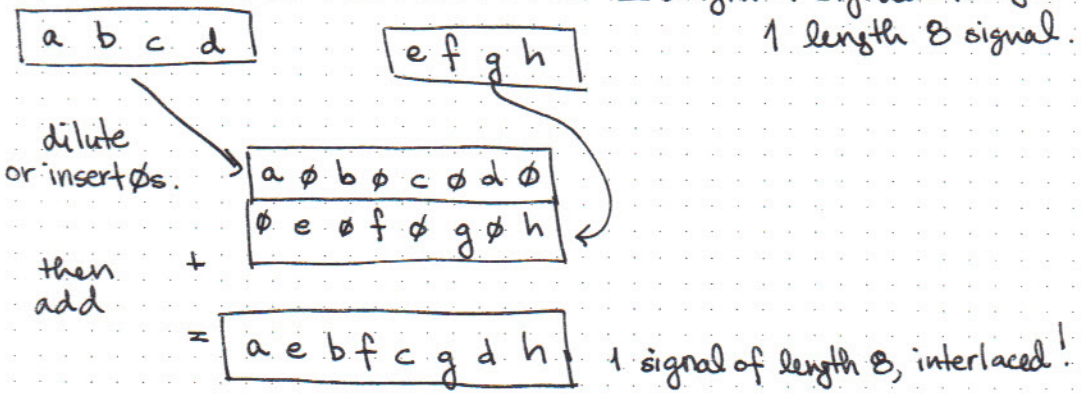
Recursive step

We need to know how to combine 2 freq. spectra of size $\ell/2$ (possibly multiplying by various factors) to get one spectrum of size ℓ.

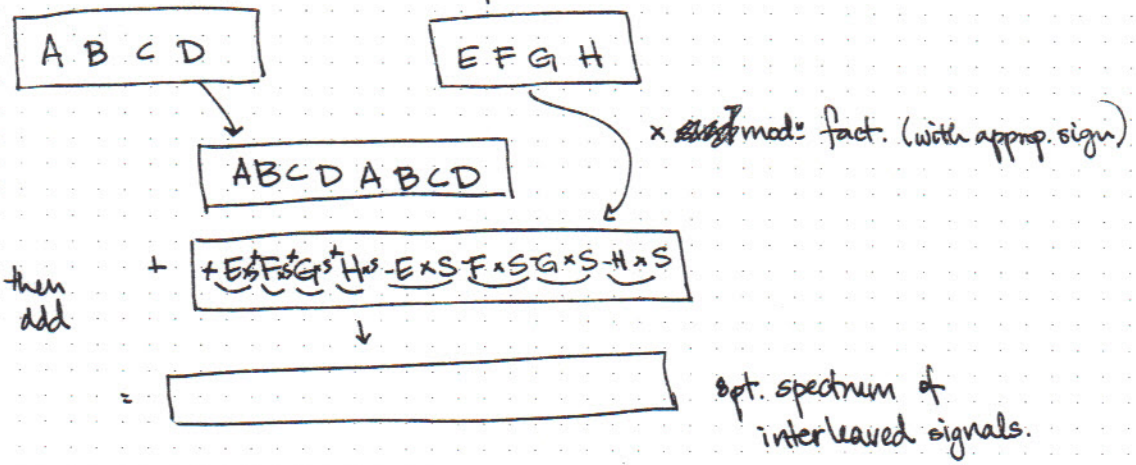
Recall that the time domain samples are from two interlaced sets and when we combine the 2 freq. spectra, we need to "undo" this interlacing.



Let's look at how to combine 2 length 4 signals to get 1 length 8 signal.



In the freq. domain, the insertion of Øs (or upsampling) corresponds to a doubling/periodization of spectrum (with a modulation to account for shift)



$A \longrightarrow A + \omega^k E$

LECTURE #4

- Def^n LINEAR APPROX^n

idea is to use a ~~linear~~ fixed (linear) subspace to approximate x.

e.g. take first K (discrete) Fourier basis vectors

Define S_K = span of ——"——

Then $x|_{S_K}^{(K)} = \underset{\uparrow}{\text{ortho}}\ \underset{\uparrow}{\text{proj}^n}$ of x onto S_K

$$= \frac{1}{\sqrt{n}} \sum_{j=0}^{K-1} \hat{x}(j)\, e^{-2\pi i j k/n}$$

i.e., just keep FIRST K Fcoeffs and do IDFT.

it's linear because S_K is a subspace

$$(x+y)|_{S_K} = x|_{S_K} + y|_{S_K}.$$

frequently used in numerical analysis ("first K Fourier modes" old-school compression?)

- Def^n NONLINEAR APPROX^n (this is sparse approx^n)

idea is to pick best K Fourier modes for a given x

↳ so as to minimize ℓ_2 error

$$x_K = \underset{\substack{\hat{z} \text{ has } K \\ \text{non zero Fcs}}}{\arg\min_{z}} \|x - z\|_2$$

Plancherel / Parseval: $\|x\|_2^2 = \sum_{k=0}^{n-1} |x(k)|^2 = \sum_{j=0}^{n-1} |\hat{x}(j)|^2 = \|\hat{x}\|_2^2$

$$\Rightarrow \|x-z\|_2^2 = \|\widehat{(x-z)}\|_2^2 = \|\hat{x} - \hat{z}\|_2^2$$

$$= \sum_{j=0}^{n-1} |\hat{x}(j) - \hat{z}(j)|^2 \qquad \hat{z} \text{ has } K \text{ non-zeros in } \Lambda_K$$

$$= \sum_{j \in \Lambda_K} |\hat{x}(j) - \hat{z}(j)|^2 + \sum_{j \notin \Lambda_K} |\hat{x}(j)|^2$$

whole expression minimized when Λ_K = set of K largest (in abs. val.) Fcoeffs of x

OVER →

$$\therefore\ x_K^{(K)} = \frac{1}{\sqrt{n}} \sum_{j \in \Lambda_K} \hat{x}(j)\, e^{-2\pi i j k/n}$$

Λ_K = K biggest Fcoeffs of x

Let's discuss how to compute the best k-term approx^{n}:

Naive algo:

- Given ~~the~~ the vector $x \in \mathbb{C}^n$,

  ① compute the FCs $\hat{x} = \mathcal{F}x$, $\hat{x} \in \mathbb{C}^n$

  ② "sort" the l.l of FCs

  ③ "retain" the k-largest (in abs. val.)

  → Discuss: How long does this take?
  How do you sort the coeffs?
  Can you do this faster/differently?
  i.e., MUST we sort all the elts of the vector in order to find the k largest?

Missing information or samples reconstruction from

- all of our discussion assumed that we had all the entries in the vector in order to compute the FT. What if we don't have all the entries?

① What are different models for sampling?

② How can we compute a specific FC with limited samples?
Are there some FCs we can't compute with certain types of samples?

③ Can we compute all of the FCs?
What if some are zero — do we need to compute them?
How do we determine which coeffs to estimate/compute?

④ What are some error metrics/models for evaluating what we can compute?

⑤ What if signal = 1 complex exponential (unknown freq). Can you find it?

⑥ What about 2 freqs?

Try to spend today in the computer lab exploring these questions. Do not resort to general machinery you might know from CS/Fourier sampling...

What we're going to talk about tomorrow combines

① reconstruction from partial information or samples of a signal

② reconstruction meaning not necessarily full signal info but a best k-term rep^n of a signal ~~that is always~~ (k-term in Fourier domain).

→ Also take a look at non-uniform FFTs! There are a number of different algos for this.