

ELEC 221 Lecture 13

The fast Fourier transform

Thursday 20 October 2022

Announcements

- Assignment 4 due next Wednesday
- Quizzes resume on Tuesday
- Tuesday's lecture will be a hands-on (instructions for what to bring will be posted on Piazza)

We introduced the **discrete-time Fourier transform** (DTFT):

Inverse DTFT (synthesis equation):

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega$$

DTFT (analysis equation):

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$$

There is a convolution property just like in CT:

$$y[n] = h[n] * x[n]$$

$$Y(e^{j\omega}) = H(e^{j\omega}) X(e^{j\omega})$$

Same consequences for the impulse response in DT:

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n] e^{-j\omega n}$$

$$h[n] = \frac{1}{2\pi} \int_{2\pi} H(e^{j\omega}) e^{j\omega n} d\omega$$

Last time

We looked at some of its key properties.

It is periodic:

$$X(e^{j(\omega+2\pi)}) = X(e^{j\omega})$$

There are some convergence criteria:

$$\sum_{n=-\infty}^{\infty} |x[n]| < \infty \quad \sum_{n=-\infty}^{\infty} |x[n]|^2 < \infty$$

Oppenheim 5.9.

Suppose $x[n]$ is a real signal with spectrum $X(e^{j\omega})$ and we know the following about it:

1. $x[n] = 0, \quad n > 0$
2. $x[0] > 0$
3. $\text{Im}(X(e^{j\omega})) = \sin \omega - \sin 2\omega$
4. $\frac{1}{2\pi} \int_{2\pi} |X(e^{j\omega})|^2 d\omega = 3$

What is $x[n]$?

$$\text{Even}(x[n]) \leftrightarrow \text{Re}(X(e^{j\omega}))$$

1. $x[n] = 0, \quad n > 0$
2. $x[0] > 0$
3. $\text{Im}(X(e^{j\omega})) = \sin \omega - \sin 2\omega$
4. $\frac{1}{2\pi} \int_{2\pi} |X(e^{j\omega})|^2 d\omega = 3$

Given a real signal $x[n]$, last class we saw that

$$\text{Odd}(x[n]) \leftrightarrow j \text{Im}(X(e^{j\omega}))$$

We can find the odd part of $x[n]$ by taking the inverse DTFT.

$$F(\text{Odd}(x[n])) = j \cdot \text{Im}(X(e^{j\omega}))$$

$$\text{Odd}(x[n]) = F^{-1}(j \cdot \text{Im}(X(e^{j\omega})))$$

$$= F^{-1}(j \sin \omega - j \sin 2\omega)$$

$$\frac{e^{j\theta} - e^{-j\theta}}{2} = \sin \theta$$

$$\text{Odd}(x[n]) = F^{-1}\left(\frac{e^{j\omega} - e^{-j\omega}}{2} - \left(\frac{e^{2j\omega} - e^{-2j\omega}}{2}\right)\right)$$

$$= \frac{1}{2} \left[F^{-1}(e^{j\omega}) - F^{-1}(e^{-j\omega}) - F^{-1}(e^{2j\omega}) + F^{-1}(e^{-2j\omega}) \right]$$

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \delta[n] e^{-j\omega n} = 1$$

$$x[n-n_0] \leftrightarrow e^{-j\omega n_0} X(e^{j\omega})$$

$$e^{j\omega} = e^{j\omega \cdot 1} \cdot 1$$

$$\delta[n+1] \leftrightarrow e^{j\omega \cdot 1} \cdot 1$$

$$\delta[n-2] \leftrightarrow e^{-2j\omega \cdot 1}$$

$$\text{Odd}(x[n]) = \frac{1}{2} \begin{bmatrix} \delta[n+1] - \delta[n-1] \\ -\delta[n+2] + \delta[n-2] \end{bmatrix}$$

Practice problem

- 1. $x[n] = 0, \quad n > 0$
- 2. $x[0] > 0$
- ✓ 3. $\text{Im}(X(e^{j\omega})) = \sin \omega - \sin 2\omega$
- 4. $\frac{1}{2\pi} \int_{2\pi} |X(e^{j\omega})|^2 d\omega = 3$

$$\text{Odd}(x[n]) = \frac{1}{2}(\delta[n+1] - \delta[n-1] - \delta[n+2] + \delta[n-2])$$

Know that

$$\text{Odd}(x[n]) = \frac{x[n] - x[-n]}{2}$$

Allows us to determine that

$$\begin{aligned} x[n] &= 2 \cdot \text{Odd}(x[n]) + x[-n] \\ x[n] &= 2 \cdot \text{Odd}(x[n]) < 0 \quad n < 0 \end{aligned}$$

Practice problem

1. $x[n] = 0$, $n > 0$
2. $x[0] > 0$
3. $\text{Im}(X(e^{j\omega})) = \sin \omega - \sin 2\omega$
4. $\frac{1}{2\pi} \int_{2\pi} |X(e^{j\omega})|^2 d\omega = 3$

So,

$$\begin{aligned} x[n] &= \delta[n+1] - \delta[n-1] - \delta[n+2] + \delta[n-2] \quad n < 0 \\ &= \delta[n+1] - \delta[n-1] - \delta[n+2] + \delta[n-2] \quad n < 0 \\ &= \delta[n+1] - \delta[n+2] \quad n < 0 \end{aligned}$$

Practice problem

- ✓ 1. $x[n] = 0, \quad n > 0$
- ✓ 2. $x[0] > 0$
- ✓ 3. $\text{Im}(X(e^{j\omega})) = \sin \omega - \sin 2\omega$
4. $\frac{1}{2\pi} \int_{2\pi} |X(e^{j\omega})|^2 d\omega = 3$

We have

$$x[n] = \begin{cases} 0 & n > 0 \\ \text{??} > 0 & n = 0 \\ \delta[n+1] - \delta[n+2] & n < 0 \end{cases}$$

Practice problem

1. $x[n] = 0, \quad n > 0$
2. $x[0] > 0$
3. $\text{Im}(X(e^{j\omega})) = \sin \omega - \sin 2\omega$
4. $\frac{1}{2\pi} \int_{2\pi} |X(e^{j\omega})|^2 d\omega = 3$

Leverage Parseval's relation:

$$\frac{1}{2\pi} \int_{2\pi} |X(e^{j\omega})|^2 d\omega = \sum_{n=-\infty}^{\infty} |x[n]|^2 = 3$$

Practice problem

1. $x[n] = 0, \quad n > 0$
- ✓ 2. $x[0] > 0$
3. $\text{Im}(X(e^{j\omega})) = \sin \omega - \sin 2\omega$
4. $\frac{1}{2\pi} \int_{2\pi} |X(e^{j\omega})|^2 d\omega = 3$

$$x[n] = \begin{cases} 0 & n > 0 \\ ?? > 0 & n = 0 \\ \delta[n+1] - \delta[n+2] & n < 0 \end{cases}$$

Structure of $x[n]$ so far means that

$$\begin{aligned} \sum_{n=-\infty}^{\infty} |x[n]|^2 &= \sum_{n=-\infty}^0 |x[n]|^2 = |x[-2]|^2 + |x[-1]|^2 + |x[0]|^2 \\ &= 1 + 1 + |x[0]|^2 \\ &= 3 \end{aligned}$$

$x[0] = \pm 1$

Practice problem

1. $x[n] = 0, \quad n > 0$
2. $x[0] > 0$
3. $\text{Im}(X(e^{j\omega})) = \sin \omega - \sin 2\omega$
4. $\frac{1}{2\pi} \int_{2\pi} |X(e^{j\omega})|^2 d\omega = 3$

$$x[n] = \begin{cases} 0 & n > 0 \\ 1 & n = 0 \\ \delta[n+1] - \delta[n+2] & n < 0 \end{cases}$$

At the end of the day, we can simply write

$$x[n] = \delta[n+1] - \delta[n+2] + \delta[n]$$



Learning outcomes:

- Distinguish between the discrete-time Fourier transform and the discrete Fourier transform
- Describe the fast Fourier transform (FFT) algorithm
- Implement a basic FFT algorithm and state its algorithmic scaling

The discrete Fourier transform

Take another look at the DTFT:

$$x[n] = \frac{1}{2\pi} \int_{2\pi} X(e^{j\omega}) e^{j\omega n} d\omega \quad X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$$

$X(e^{j\omega})$ is a **continuous function** of ω .

The discrete Fourier transform

Consider what we have been doing in Python so far:

```
>>> import numpy as np
>>> x = np.array([1, 2, 3, 4, 5])
>>> y = np.fft.fft(x)
>>> y
array([15. +0.j,
       -2.5+3.4409548j,
       -2.5+0.81229924j,
       -2.5-0.81229924j,
       -2.5-3.4409548j])
```

We are computing a Fourier spectrum here. The signal is discrete and finite, and *the spectrum is also discrete and finite*. What is this thing?

The discrete Fourier transform

Recall how we derived the DTFT. We had some signal, and considered a periodic extension of it.

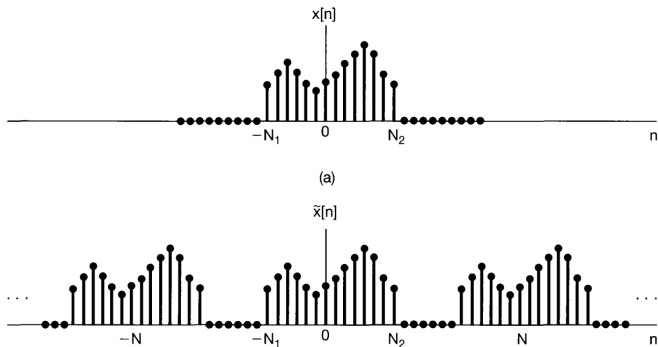
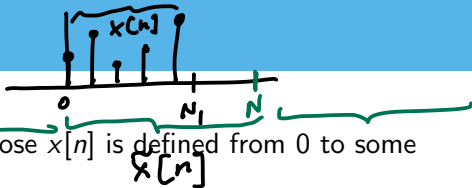


Image credit: Oppenheim chapter 5.1

The discrete Fourier transform



Without loss of generality, suppose $x[n]$ is defined from 0 to some $N_1 \leq N$.

We can define $\tilde{x}[n]$ as a periodic function with period N . Then, it has Fourier series coefficients

$$\begin{aligned} c_k &= \frac{1}{N} \sum_{n \in \langle N \rangle} \tilde{x}[n] e^{-jk \frac{2\pi n}{N}} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-jk \frac{2\pi n}{N}} \end{aligned}$$

These $N - 1$ coefficients are known as the **discrete Fourier transform** of $x[n]$.

The discrete Fourier transform

Generally we write

$$\tilde{X}[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-jk \frac{2\pi n}{N}} \quad k=0, 1, \dots, N-1$$

DTFT: $\sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$

Why are these interesting?

First, we can recover $x[n]$ from them:

$$x[n] = \sum_{k=0}^{N-1} \tilde{X}[k] e^{jk \frac{2\pi n}{N}} \quad n=0, \dots, N-1$$

The discrete Fourier transform

More importantly, these are related to the discrete-time Fourier transform:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$$

If the signal is only defined from 0 to $N - 1$,

$$X(e^{j\omega}) = \sum_{n=0}^{N-1} x[n] e^{-j\omega n}$$

The discrete Fourier transform

$$x(e^{j\omega}), \quad \omega = k \cdot \frac{2\pi}{N}$$

What if we sample this signal at particular values of $k\omega = k2\pi/N$?

$$X(e^{j \cdot k \cdot \frac{2\pi}{N}}) = \sum_{n=0}^{N-1} x[n] e^{-j \cdot k \cdot \frac{2\pi n}{N}} = N \cdot \tilde{X}[k]$$

$$\frac{1}{N} X(e^{j k \cdot \frac{2\pi}{N}}) = \tilde{X}[k]$$

The discrete Fourier transform is a set of equally spaced samples of the full discrete-time Fourier transform.

Key point 1: Any signal $x[n]$ can be uniquely specified by a finite set of samples from its DTFT (i.e., its DFT).

The fast Fourier transform

How do we actually compute the DFT?

$$\underbrace{\tilde{X}[k]}_{k=0, \dots, N-1} = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-jk \cdot \frac{2\pi n}{N}}$$

$$\tilde{X}[0] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] = \frac{1}{N} (x[0] + x[1] + \dots + x[N-1])$$

$$\tilde{X}[1] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j \cdot \frac{2\pi n}{N}} = \frac{1}{N} (x[0] + x[1] e^{-j \cdot \frac{2\pi}{N}} + \dots)$$

\vdots

$$\tilde{X}[N-1] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j(N-1)2\pi n/N} = \frac{1}{N} (x[0] + x[1] e^{-j(N-1)2\pi/N} + \dots)$$

The fast Fourier transform

This is just a big matrix multiplication!

$$\begin{bmatrix} \tilde{X}[0] \\ \tilde{X}[1] \\ \vdots \\ \tilde{X}[N-1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{j\frac{2\pi}{N}} & e^{j2\frac{2\pi}{N}} & \cdots & e^{j(N-1)\frac{2\pi}{N}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-j(N-1)\frac{2\pi}{N}} & e^{-j2(N-1)\frac{2\pi}{N}} & \cdots & e^{-j(N-1)^2\frac{2\pi}{N}} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

The fast Fourier transform

We typically define $\zeta = e^{-2\pi j/N}$ (N th root of unity):

$$\begin{bmatrix} \tilde{X}[0] \\ \tilde{X}[1] \\ \vdots \\ \tilde{X}[N-1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \zeta & \zeta^2 & \dots & \zeta^{N-1} \\ 1 & \zeta^2 & \zeta^4 & \dots & \zeta^{2(N-1)} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & \zeta^{N-1} & \zeta^{2(N-1)} & \dots & \zeta^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

The fast Fourier transform

$$O(N^2)$$

This matrix is very beautiful:

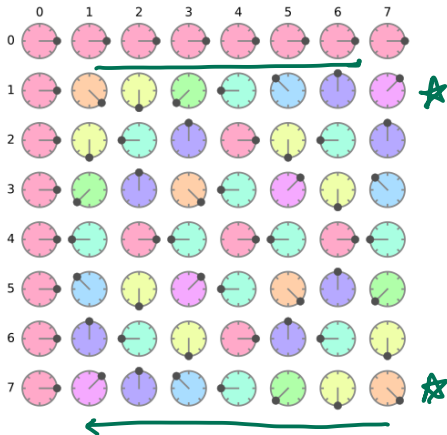


Image credit: Darnling - Own work. CCBY-SA4.0 <https://creativecommons.org/licenses/by-sa/4.0>

The fast Fourier transform

Key point 2: There is an efficient algorithm for the DFT: the **fast** Fourier transform.

$$\tilde{X}[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-jk \cdot \frac{2\pi n}{N}}$$

$$x[n] = \sum_{k=0}^{N-1} \tilde{X}[k] e^{jk \cdot \frac{2\pi n}{N}}$$

These correspond (almost) to:

```
>>> X_k = np.fft.fft(xn)
>>> xn = np.fft.ifft(X_k)
```

The fast Fourier transform

Note that in NumPy, the normalization convention is “backwards”:

Implementation details

There are many ways to define the DFT, varying in the sign of the exponent, normalization, etc. In this implementation, the DFT is defined as

$$A_k = \sum_{m=0}^{n-1} a_m \exp\left\{-2\pi i \frac{mk}{n}\right\} \quad k = 0, \dots, n-1.$$

To match the convention we use in class, you need to specify:

```
>>> y = np.fft.fft(x, norm='forward')
```

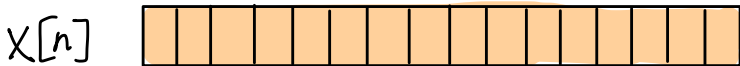
Screenshot: <https://numpy.org/doc/stable/reference/routines.fft.html#background-information>

The fast Fourier transform

The most famous FFT algorithm is the **Cooley-Tukey algorithm**: it computes the DFT using a divide and conquer method.

Nicest case to analyze is when N is a power of 2.

Suppose we would like to compute the k th element of the DFT.



$$\tilde{X}[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi k n}{N}}$$

The fast Fourier transform

$$x[n] \quad \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \boxed{} \boxed{}$$

$$\tilde{X}[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi k n}{N}}$$

$$x[n] \quad \boxed{\text{yellow}} \boxed{\text{pink}} \boxed{\text{yellow}} \boxed{\text{pink}} \boxed{\text{yellow}} \boxed{\text{pink}} \boxed{\text{yellow}} \boxed{\text{pink}} \boxed{\text{yellow}} \boxed{\text{pink}} \boxed{\text{yellow}} \boxed{\text{pink}} \boxed{\text{yellow}} \boxed{\text{pink}} \boxed{\text{yellow}} \boxed{\text{pink}}$$

$$\tilde{X}[k] = \sum_{m=0}^{\frac{N}{2}-1} x[2m] e^{-j \frac{2\pi k (2m)}{N}} + \sum_{m=0}^{\frac{N}{2}-1} x[2m+1] e^{-j \frac{2\pi k (2m+1)}{N}}$$

$$= \sum_{m=0}^{\frac{N}{2}-1} x[2m] e^{-j \frac{2\pi k (2m)}{N}} + e^{-j \frac{2\pi k}{N}} \sum_{m=0}^{\frac{N}{2}-1} x[2m+1] e^{-j \frac{2\pi k (2m)}{N}}$$

The fast Fourier transform



$$\tilde{X}[k] = \tilde{E}[k] + e^{-j\frac{2\pi k}{N}} \cdot \tilde{O}[k]$$

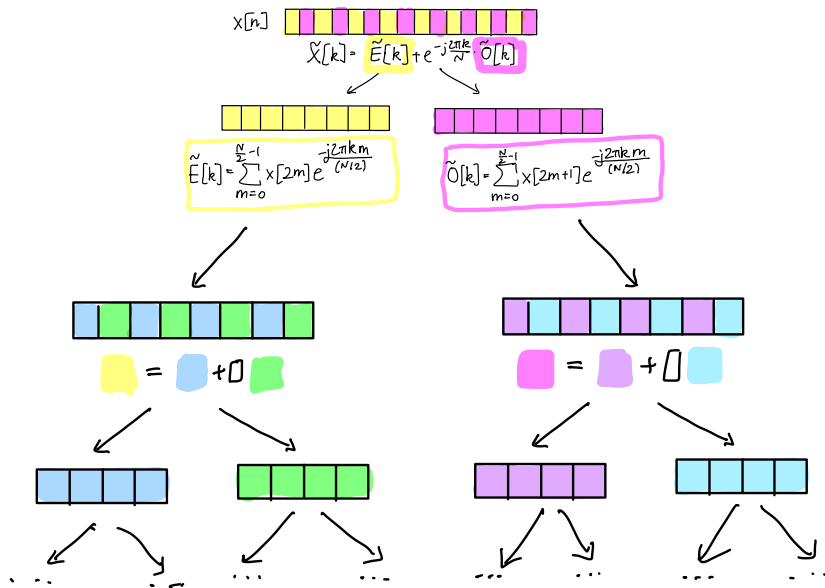


$$\tilde{E}[k] = \sum_{m=0}^{\frac{N}{2}-1} x[2m] e^{-j\frac{2\pi k m}{(N/2)}}$$

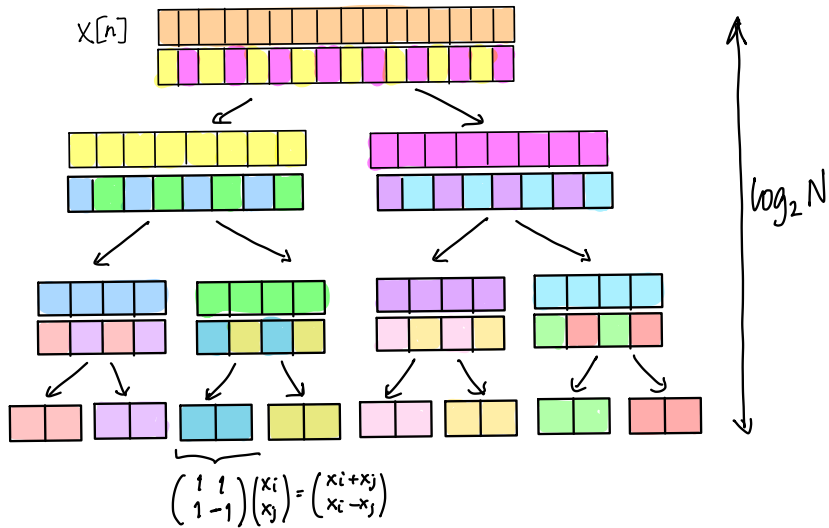


$$\tilde{O}[k] = \sum_{m=0}^{\frac{N}{2}-1} x[2m+1] e^{-j\frac{2\pi k m}{(N/2)}}$$

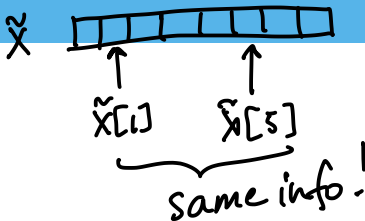
The fast Fourier transform



The fast Fourier transform



The fast Fourier transform



But wait!

We get something for free.

$$\begin{aligned}
 \tilde{X}[k] &= \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{N} kn} \\
 &= \sum_{m=0}^{N/2-1} x[2m] e^{-j \frac{2\pi k}{N} \cdot 2m} + e^{-j \frac{2\pi k}{N} \cdot N/2} \sum_{m=0}^{N/2-1} x[2m+1] e^{-j \frac{2\pi k}{N} \cdot 2m} \\
 &= \tilde{E}[k] + e^{-j \frac{2\pi k}{N} \cdot N/2} \tilde{O}[k]
 \end{aligned}$$

$$\tilde{X}[k + N/2] \rightarrow \tilde{X}[k]$$

The fast Fourier transform

Recall that there is some symmetry between positive/negative values in Fourier coefficients.

$$\begin{aligned}
 \tilde{X}\left[k + \frac{N}{2}\right] &= \sum_{m=0}^{N/2-1} x[2m] e^{-j \frac{2\pi}{N} \left(k + \frac{N}{2}\right) \cdot m} \\
 &= \sum_{m=0}^{N/2-1} x[2m] e^{-j \frac{2\pi}{N} k m} e^{-j \frac{2\pi}{N} \left(\frac{N}{2}\right) m} \\
 &= \sum_{m=0}^{N/2-1} x[2m] e^{-j \frac{2\pi}{N} k m} e^{-j \pi m} \\
 &= \sum_{m=0}^{N/2-1} x[2m] e^{-j \frac{2\pi}{N} k m} (-1)^m \\
 &= \sum_{m=0}^{N/2-1} x[2m] e^{-j \frac{2\pi}{N} k m} e^{-j \pi m} \\
 &= \sum_{m=0}^{N/2-1} x[2m] e^{-j \frac{2\pi}{N} k m} (-1)^m
 \end{aligned}$$

The fast Fourier transform

$$\begin{aligned}\tilde{X}[k + N/2] &= \sum_{m=0}^{N/2-1} x[2m] e^{-j \frac{2\pi}{N/2} km} e^{-j 2\pi m} \\ &\quad + e^{-j(k+N/2) \frac{2\pi}{N}} \sum_{m=0}^{N/2-1} x[2m+1] e^{-j \frac{2\pi}{N/2} km} e^{-j 2\pi m} \\ &= \sum_{m=0}^{N/2-1} x[2m] e^{-j \frac{2\pi}{N/2} km} \\ &\quad + e^{-j(k+N/2) \frac{2\pi}{N}} \sum_{m=0}^{N/2-1} x[2m+1] e^{-j \frac{2\pi}{N/2} km}\end{aligned}$$

The fast Fourier transform

$$\begin{aligned}\tilde{X}[k + N/2] &= \sum_{m=0}^{N/2-1} x[2m] e^{-j \frac{2\pi}{N/2} km} \\ &\quad + e^{-jk \frac{2\pi}{N}} e^{-j\pi} \sum_{m=0}^{N/2-1} x[2m+1] e^{-j \frac{2\pi}{N/2} km} \\ &= \tilde{E}[k] - e^{jk \cdot \frac{2\pi}{N}} \tilde{O}[k] \\ &= \sum_{m=0}^{N/2-1} x[2m] e^{-j \frac{2\pi}{N/2} km} \\ &\quad - e^{-jk \frac{2\pi}{N}} \sum_{m=0}^{N/2-1} x[2m+1] e^{-j \frac{2\pi}{N/2} km} \\ &= \tilde{E}[k] - e^{-jk \frac{2\pi}{N}} \tilde{O}[k]\end{aligned}$$

The fast Fourier transform

Only need to compute the first 1/2 of values in each subdivision of the problem.

$$\tilde{X}[k] = \tilde{E}[k] + e^{-jk\frac{2\pi}{N}} \tilde{O}[k]$$

$$\tilde{X}[k + \frac{N}{2}] = \tilde{E}[k] - e^{-jk\frac{2\pi}{N}} \tilde{O}[k]$$

At the end of the day, the complexity of the FFT is $O(N \log_2 N)$ - way better than $O(N^2)$ matrix multiplication!

The fast Fourier transform

Let's program it. We will compare:

- The naive matrix multiplication version
- A naive implementation of the FFT
- NumPy's FFT

If you want to dig deeper:

<https://nbviewer.org/url/jakevdp.github.io/downloads/notebooks/UnderstandingTheFFT.ipynb>

Today's learning outcomes were:

- Distinguish between the discrete-time Fourier transform and the discrete Fourier transform
- Describe the steps of the fast Fourier transform (FFT) algorithm
- Implement a basic FFT algorithm and state its algorithmic scaling

What topics did you find unclear today?

For next time

Content:

- Hands-on: 2D Fourier analysis and image processing

Action items:

1. Quiz 6 on Tuesday
2. Assignment 4 due on Wednesday
3. Check Piazza for instructions on what to bring for next class

Recommended reading:

- From today's class: Oppenheim extension problems 5.53-5.54
- For next class: research the 2D DFT