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)

Last time

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

Inverse DTFT (synthesis equation):

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

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:

$$\begin{aligned} y[n] &= h[n] * x[n] \\ Y(e^{j\omega}) &= H(e^{j\omega})X(e^{j\omega}) \end{aligned}$$

Same consequences for the impulse response in DT:

$$\begin{aligned} H(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega n} \\ h[n] &= \frac{1}{2\pi}\int_{2\pi} H(e^{j\omega})e^{j\omega n}d\omega \end{aligned}$$

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:

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

Practice problem

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. $\mathrm{Im}(X(e^{j\omega})) = \sin\omega - \sin 2\omega$
4. $\frac{1}{2\pi}\int_{2\pi} |X(e^{j\omega})|^2 d\omega = 3$

What is $x[n]$?

Practice problem

1. $x[n] = 0, \quad n > 0$
2. $x[0] > 0$
3. $\mathrm{Im}(X(e^{j\omega})) = \sin\omega - \sin 2\omega$
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

$$\mathrm{Odd}(x[n]) \overset{\mathcal{F}}{\longleftrightarrow} j\mathrm{Im}(X(e^{j\omega}))$$

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

Practice problem

$$\begin{aligned}
\mathrm{Odd}(x[n]) &= \mathcal{F}^{-1}\left(j\mathrm{Im}(X(e^{j\omega}))\right) \\
&= \mathcal{F}^{-1}\left(j\sin\omega - j\sin 2\omega\right) \\
&= \mathcal{F}^{-1}\left(\frac{1}{2}(e^{j\omega} - e^{-j\omega} - e^{2j\omega} + e^{-2j\omega})\right) \\
&= \frac{1}{2}(\delta[n+1] - \delta[n-1] - \delta[n+2] + \delta[n-2])
\end{aligned}$$

Practice problem

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

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

Know that

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

Allows us to determine that

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

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. $\frac{1}{2\pi}\int_{2\pi} |X(e^{j\omega})|^2 d\omega = 3$

So,

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

Practice problem

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

We have

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

Practice problem

1. $x[n] = 0, \quad n > 0$
2. $x[0] > 0$
3. $\mathrm{Im}(X(e^{j\omega})) = \sin\omega - \sin 2\omega$
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. $\mathrm{Im}(X(e^{j\omega})) = \sin\omega - \sin 2\omega$
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

$$\sum_{n=-\infty}^{\infty} |x[n]|^2 = \sum_{n=-\infty}^{0} |x[n]|^2 = |x[0]|^2 + 2 = 3$$

$$|x[0]|^2 = 1 \rightarrow x[0] = 1$$

Practice problem

1. $x[n] = 0, \quad n > 0$
2. $x[0] > 0$
3. $\mathrm{Im}(X(e^{j\omega})) = \sin\omega - \sin 2\omega$
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

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

Today

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

Take another look at the DTFT:

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

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

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.

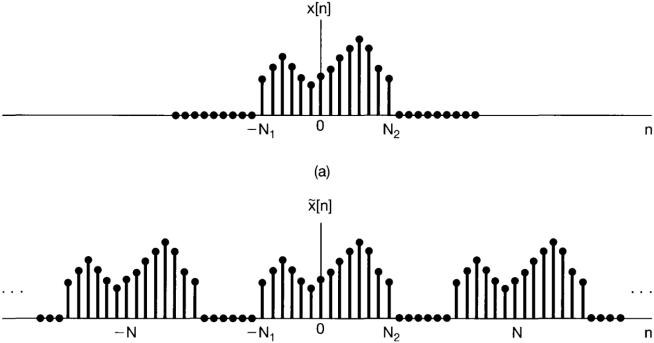


Image credit: Oppenheim chapter 5.1

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

$$\begin{aligned} c_k &= \frac{1}{N} \sum_{n=\langle N \rangle} \tilde{x}[n] e^{-jk2\pi n/N} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-jk2\pi n/N} \end{aligned}$$

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

Generally we write

$$\tilde{X}[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-jk2\pi n/N}, \quad k = 0, \ldots N-1$$

Why are these interesting?

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

$$x[n] = \sum_{k=0}^{N-1} \tilde{X}[k] e^{jk2\pi n/N}, \quad n = 0, \ldots N-1$$

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 $N-1$,

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

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

$$
\begin{aligned}
X(e^{jk2\pi/N}) &= \sum_{n=0}^{N-1} x[n]e^{-jk2\pi n/N} = N\tilde{X}[k] \\
\frac{1}{N}X(e^{jk2\pi/N}) &= \tilde{X}[k]
\end{aligned}
$$

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).

How do we actually compute the DFT?

$$\tilde{X}[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk2\pi n/N}$$

$$\begin{aligned}
\tilde{X}[0] &= \frac{1}{N}\sum_{n=0}^{N-1} x[n] = \frac{1}{N}\left(x[0] + x[1] + \cdots + x[N-1]\right) \\
\tilde{X}[1] &= \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk2\pi n/N} = \frac{1}{N}\left(x[0] + x[1]e^{-j2\pi/N} + \cdots\right) \\
&\vdots \\
\tilde{X}[N-1] &= \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j(N-1)2\pi n/N} = \frac{1}{N}\left(x[0] + x[1]e^{-j(N-1)2\pi/N} + \cdots\right)
\end{aligned}$$

The fast Fourier transform

This is just a big matrix multiplication!

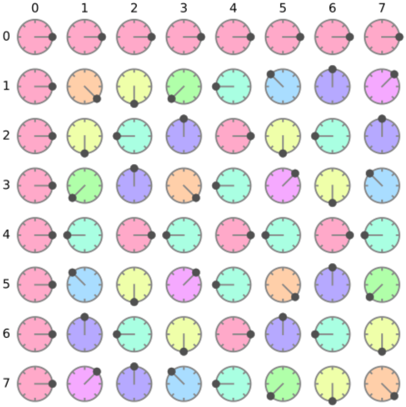
$$\begin{bmatrix} \tilde{X}[0] \\ \tilde{X}[1] \\ \dots \\ \tilde{X}[N-1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-j2\pi/N} & \cdots & e^{-j2\pi(N-1)/N} \\ \vdots & \ddots & \ddots & \vdots \\ 1 & e^{-j(N-1)2\pi/N} & \dots & e^{-j(N-1)^2 2\pi/N} \end{bmatrix} \begin{bmatrix} x[0]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

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

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

The fast Fourier transform

This matrix is very beautiful:



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

$$
\begin{aligned}
\tilde{X}[k] &= \frac{1}{N}\sum_{n=0}^{N-1} x[n] e^{-jk2\pi n/N} \\
x[n] &= \sum_{k=0}^{N-1} \tilde{X}[k] e^{jk2\pi n/N}
\end{aligned}
$$

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. Ir this implementation, the DFT is defined as

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

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

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

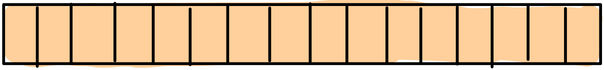
Screenshot: `https://numpy.org/doc/stable/reference/routines.fft.html#background-information`

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

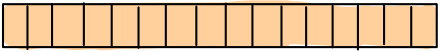
Nicest case to analyze is when N is a power of 2.

Suppose we would like to compute the kth element of the DFT.

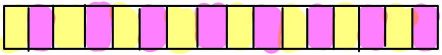
$x[n]$



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

$x[n]$ 

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

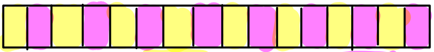
$x[n]$ 

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

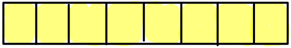
$$= \sum_{m=0}^{\frac{N}{2}-1} x[2m]\, e^{-j\frac{2\pi k(2m)}{N}} + e^{-j\frac{2\pi k}{N}} \sum_{m=0}^{\frac{N}{2}-1} x[2m+1]\, e^{-j\frac{2\pi k(2m)}{N}}$$

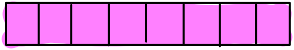
The fast Fourier transform

$x[n]$



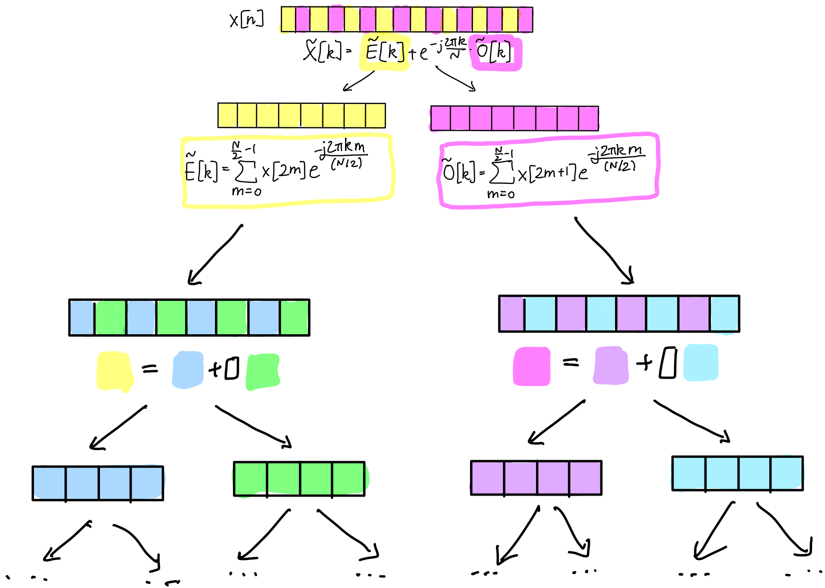
$$\tilde{X}[k] = \boxed{\tilde{E}[k]} + e^{-j\frac{2\pi k}{N}} \cdot \boxed{\tilde{O}[k]}$$



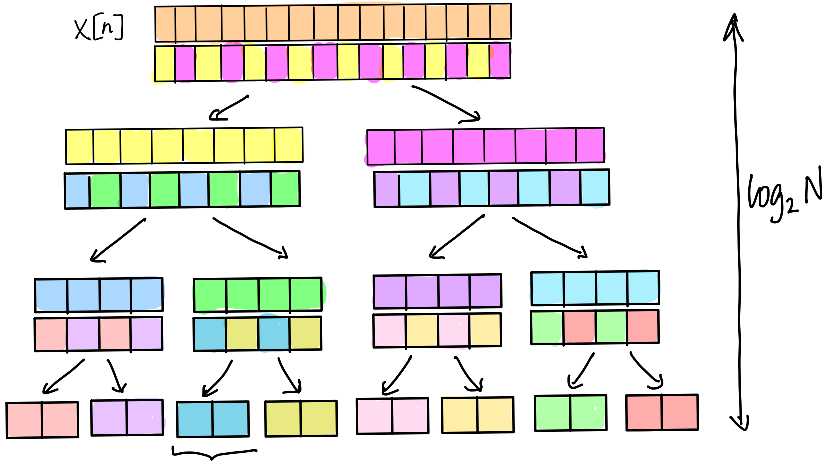


$$\boxed{\tilde{E}[k] = \sum_{m=0}^{\frac{N}{2}-1} x[2m]\, e^{-j\frac{2\pi k m}{(N/2)}}}$$

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


$x[n]$
$\tilde{X}[k] = \tilde{E}[k] + e^{-j\frac{2\pi k}{N}} \cdot \tilde{O}[k]$
$\tilde{E}[k] = \sum_{m=0}^{\frac{N}{2}-1} x[2m] e^{\frac{-j2\pi km}{(N/2)}}$
$\tilde{O}[k] = \sum_{m=0}^{\frac{N}{2}-1} x[2m+1] e^{\frac{-j2\pi km}{(N/2)}}$

The fast Fourier transform



$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_i \\ x_j \end{pmatrix} = \begin{pmatrix} x_i + x_j \\ x_i - x_j \end{pmatrix}$$

But wait!

We get something for free.

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

The fast Fourier transform

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

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

The fast Fourier transform

$$
\begin{aligned}
\tilde{X}[k + N/2] &= \sum_{m=0}^{N/2-1} x[2m] e^{-j\frac{2\pi}{N/2}km} e^{-j2\pi m} \\
&\qquad + e^{-j(k+N/2)\frac{2\pi}{N}} \sum_{m=0}^{N/2-1} x[2m+1] e^{-j\frac{2\pi}{N/2}km} e^{-j2\pi m}
\end{aligned}
$$

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

The fast Fourier transform

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

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

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

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

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!

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