What is 6.003?

What is a signal?
Abstactly, a signal is a function that conveys information.

Signal processing is about extracting meaningful information from
signals, and/or manipulating information in signals to produce new
signals.

What is a transform?
Provide multiple views/perspectives on a signal
Some information more clearly visible (and/or more easily manipu-
lable) from one perspective than another.

Why Fourier?
One reason: Many aspects of human perception are related to fre-
quency representation.

Some things apparent in frequency but not in time (and vice versa).
Today: Lossy Compression

As opposed to “lossless” compression (LZW, Huffman, zip, gzip, xzip, ...), “lossy” compression achieves a decrease in file size by throwing away information from the original signal.

Examples: JPEG, MP3

Goal: convey the “important” parts of the signal using as few bits as possible.

Check Yourself!

You want to send a song to a friend. You have the song in CD-quality WAV format (16 bits per sample, 44.1kHz sampling rate).

If the song is 3 minutes and 24 seconds long, how many bits will you have to send in order to transmit the song?

What are some examples of techniques you could use to send (roughly) the same song, but with fewer bits?

Lossy Compression

Key idea: throw away the “unimportant” bits (i.e., bits that won’t be noticed). Doing this involves knowing something about what it means for something to be noticeable.

Many aspects of human perception are frequency based → many lossy formats use frequency-based methods (along w/ models of human perception).
Lossy Compression: High-level View

To Encode:
- Split signal into “frames”
- Transform each frame into Fourier representation
- Throw away (or attenuate) some coefficients
- Additional lossless compression (LZW, RLE, Huffman, etc.)

To Decode:
- Undo lossless compression
- Transform each frame into time/spatial representation

This is pretty standard! Both JPEG and MP3, for example, work roughly this way.

Given this, one goal is to get the “important” information in a signal into relatively few coefficients in FD (“energy compaction”).

Energy Compaction

One goal is to get the “important” information in a signal into relatively few coefficients in FD (“energy compaction”).

It turns out the DFT has some problems in this regard. Consider the following signal, broken into 8-sample-long frames:

Why is the DFT undesirable in this case, given our goal of compression?

Discrete Cosine Transform

It is much more common to use the DCT (Discrete Cosine Transform) in compression applications. The DCT (or variants thereof) are used in JPEG, AAC, Vorbis, WMA, MP3, ....

The DCT (more formally, the DCT-II) is defined by:

\[ X_C[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \cos \left( \frac{\pi}{N} \left( n + \frac{1}{2} \right) k \right) \]
DCT: Relationship to DFT

\[
X_C[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \cos \left( \frac{\pi}{N} \left( n + \frac{1}{2} \right) k \right)
\]

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

\[
= \frac{1}{2N} e^{-j \frac{\pi}{4} k} \sum_{n=0}^{N-1} x[n] \left( e^{\frac{j \pi}{2N} (n+\frac{1}{2})k} + e^{-j \frac{\pi}{2N} (n+\frac{1}{2})k} \right)
\]

\[
= \frac{1}{2N} e^{-j \frac{\pi}{4} k} \left( \sum_{n=0}^{N-1} x[n] e^{-j \frac{\pi}{2N} nk} - \sum_{n=0}^{N-1} x[n] e^{-j \frac{\pi}{2N} nk} \right)
\]

\[
= \frac{1}{2N} e^{-j \frac{\pi}{4} k} \left( \sum_{n=-N}^{N-1} \tilde{x}[n] e^{-j \frac{\pi}{2N} nk} \right) = \left( e^{-j \frac{\pi}{4} k} \right) \tilde{X}[k]
\]

where \( \tilde{x}[\cdot] \) is given by the following, and the DFT coefficients \( \tilde{X}[\cdot] \) are computed with an analysis window of length \( 2N \):

\[\tilde{x}[n] = \begin{cases} x[n] & \text{if } 0 \leq n < N \\ x[-n-1] & \text{if } -N < n < 0 \end{cases}\]

The Discrete Cosine Transform

The DCT is commonly used in compression applications.

We can think about computing the DCT by first putting a mirrored copy of a windowed signal next to itself, and then computing the DFT of that new signal (shifted by 1/2 sample):

8 sample "frame"

\[
\begin{array}{cccccccc}
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
\end{array}
\]

16-sample shifted, mirrored frame

\[
\begin{array}{cccccccc}
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
\end{array}
\]

Why is the DCT more appropriate, given our goals? How does this approach fix the issue(s) we saw with the DFT?
**Energy Compaction Example: Ramp**

For many authentic signals (photographs, etc), the DCT has good "energy compaction": most of the energy in the signal is represented by relatively few coefficients.

Consider DFT vs DCT of a "ramp."

---

**A Real Example: JPEG**

To Encode:
- Color encoding (RGB → YCrCb)
- 2D-DCT of 8-by-8 blocks
- Quantization
- Run-length encoding and Huffman encoding

We'll start with something related, but not actually JPEG, then we'll refine things.

---

**Not-quite-JPEG**

To start, let's do:
- Monochrome images only
- 2D-DCT of 8-by-8 blocks
- In each block, zero out coefficients that are below some threshold
- No lossless compression on top
**DCT vs DFT**

Consider two variants of this algorithm:

- Monochrome images only
- 2D-DCT of 8-by-8 blocks
- In each block, zero out coefficients that are below some threshold
- No lossless compression on top

versus:

- Monochrome images only
- 2D-DFT of 8-by-8 blocks
- In each block, zero out coefficients that are below some threshold
- No lossless compression on top

Start by comparing DCT vs DFT: where is the energy?

---

**Energy Compaction**

Consider one 8×8 block from an image:

---

**Quantitative Comparison**

For this experiment, we'll use a measure known as the Peak Signal to Noise Ratio (PSNR) as an objective assessment of image quality. The PSNR is defined as:

\[
\text{PSNR} = 20 \times \log_{10}(255) - 10 \times \log_{10}(\text{MSE})
\]

where MSE represents the mean squared error:

\[
\text{MSE} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (I(i,j) - K(i,j))^2
\]

where \(I\) is our input image, and \(K\) is the result of our compression.
DCT vs DFT: PSNR vs fraction of coefficients kept

For the image from the previous slide, try removing different amounts of DCT (or FFT) coefficients, and measure PSNR of reconstructed image:

```
Percentage of Coefficients Kept
20
30
40
50
60

PSNR
DCT
DFT

Keeping 50% of FFT coefficients results in a PSNR ~21.8. We can achieve the same PSNR when keeping only ~35% of DCT coefficients!
```

---

Toward Authentic JPEG: Quantization

Typically, we don’t just throw away low-energy coefficients. We use some kind of a model of human perception to boost/attenuate certain frequencies. In JPEG, the actual transformation involves a point-by-point division with a quantization table, for example:

```
q[k_r, k_c] → k_c

16 11 10 16 24 40 51 61
12 12 14 19 26 58 60 55
14 13 16 24 40 57 69 56
```

---

Example: Closer to Actual JPEG

Let’s look at something closer to real JPEG.