FFT & Signal Processing (1 & 2 - dim.)

Closer look at:
- Sampling/discretization 1/2-dim.
- Fourier Representation/Frequency domain

Sampling: We've used grids for all the differential equations, think about continuous functions as sums of sinusoids, and sample:

Sampling relatively fast

... OK

Sampling too slowly

... Not OK

We are deceived into thinking this a lower frequency. A high frequency is masquerading as a lower one. We say the lower frequency is an alias of the higher.
As usual, it's more illuminating to view sinusoids as projections of points moving around circles.

**Phasors, complex exponential representation**

- We used this in Von Neumann's stability analysis.
- We used this to solve wave eqn. for string in separation of variables.

\[
\begin{align*}
\omega & \text{ radians/sec} \\
f & \text{ cycles/sec} = \frac{\omega}{2\pi} \\
T & \text{ period} = \frac{1}{f} = \frac{2\pi}{\omega} \text{ sec/cycle}
\end{align*}
\]

Sampling fast enough

not sampling fast enough

Condition for unambiguous resolution of frequency is

\[ T_s \leq \frac{T}{2} \]

\[ f_s \geq 2f \]

Appears to going backward!
Nyquist's criterion: Must sample at a rate at least twice the highest frequency in signal.

Put another way, if we sample at rate \( ws \), the highest allowed signal frequency is

\[ ws/2 = \text{"Nyquist frequency"} \]

When sampling at freq. \( ws \), we therefore can consider all frequencies as lying between \(-ws/2\) and \(+ws/2\):

\[ \text{negative frequencies} \iff \cos wt = \frac{1}{2} e^{jwt} + \frac{1}{2} e^{-jwt} \]

Yet another way to look at aliasing: All frequencies that differ by an integer multiple of \( ws \) radians/Sec are indistinguishable.
Aliasing:

audio aliasing is very disturbing because harmonic components get aliased to non-harmonic components dealt with by pre-filting, removing components above \( \frac{w_s}{2} \) before sampling

Images aliasing often shows up as disturbing herringbone patterns (Moire patterns).

For example, striped shirt on TV raster scan

Note: the frequency axis after sampling can therefore also be thought of as a circle:
Fourier Analysis

Recall vibrating string: modes

... etc.

the shape at any time is a linear combination of these sinusoids. This is a general principle - any waveform can be represented as a sum of sinusoids.

Sampled version is called the Discrete Fourier Transform.

finite sampled signal

Set of frequencies used for representation

think of samples' domain and frequency domain on circles

Nyquist Frequency
Algebraically:

Fourier Representation

\[ \chi_t = \frac{1}{N} \sum_{k=0}^{N-1} \chi_k e^{i(k2\pi/N)t} \]

amount of frequency \( k \cdot \frac{2\pi}{N} \)

= spectral component
= Fourier component

How to find Fourier components:

\( N \) equals \( N \) unknown \( \rightarrow \) uniquely determined.

[If you know linear algebra, the frequency components \( e^{j(k2\pi/N)t} \) form an orthogonal basis]

turns out that

\[ \chi_k = \sum_{t=0}^{N-1} \chi_t e^{-j(k2\pi/N)t} \]

Forward DFT

\[ \chi_t = \frac{1}{N} \sum_{k=0}^{N-1} \chi_k e^{j(k2\pi/N)t} \]

Inverse DFT

the \( \frac{1}{N} \) shows up one place or another; conventionally we put it in the inverse DFT

typically, we need 1024 or 2048 to get a good frequency representation.
Oversampling: Idea:

1) Sample much faster than necessary

\[ \omega \]

Very high Nyquist frequency

Very little aliasing

3) Then filter out components above the final Nyquist frequency, using cheap digital filtering.

\[ \omega \]

final Nyquist frequency

removed with digital filter

3) Then reduce sampling rate (sub-sample) to final, practical rate.

That's analog-to-digital conversion.

A similar idea is used in CD-players on digital-to-analog conversion:

D) Increase the Nyquist rate in the digital domain (by inserting zeros)

\[ \omega \]

\[ \omega \]

2) Digital filter

\[ \omega \]

3) Convert at higher rate \( \Rightarrow \) much less aliasing
Naïve algorithm \[ N \text{ points } \times N \text{ multiplication per point} \]

Divide & Conquer (like merge sort)

1) Divide sequence into two parts
2) FFT each subsequence
3) merge in linear time

time for \( N \)-pt. transform

\[
T(N) = aT(N/2) + cN
\]

recursive
merge time
cells
to half-sized
problems

\[
T(N) = cN + 2 \left[ c \frac{N}{2} + 2 \left[ c \frac{N}{4} + 2 \left[ c \frac{N}{8} + \ldots \right] \right] \right] \text{ log}_2 N \text{ stages}
\]

\[
T(N) = O(N \log N)
\]

Decimation-in-time algorithm divides sequence into even- and odd-numbered subsequences. Requires "shuttle" to reassemble.

Merge step look like 

"butterfly"