Review from last lecture:

- A discrete signal still has a continuous Fourier:
  \[ h[n] \rightarrow \sum_{n=0}^{L-1} h[n]e^{-j\omega n} \]
  \[ \omega = \frac{2\pi s}{L} \]
  \[ \text{DFT: } \omega = \frac{2\pi k}{L} \rightarrow H[k] \]

- Remember the effect of derivation in Fourier domain:
  \[ \frac{df(t)}{dt} \leftrightarrow j\omega F(\omega) \]

Short Matlab tutorial

- fft of a \( \cos(2\pi t/8) \): two spikes in the Fourier domain:

\[
\begin{align*}
\cos(2\pi t/8) & \rightarrow \delta(\omega - \frac{2\pi}{8}) + \delta(\omega + \frac{2\pi}{8}) \\
\text{cont. FT} & \rightarrow \frac{1}{2}(\delta(\omega - \frac{2\pi}{8}) + \delta(\omega + \frac{2\pi}{8})) \\
\text{discrete FT} & \rightarrow \omega = \frac{2\pi k}{L} = \frac{2\pi}{32}
\end{align*}
\]

\[ \omega = \frac{2\pi}{8}, \quad \frac{2\pi k}{32} = \frac{2\pi}{8}, \quad k = 4 \]
\[ \omega = -\frac{2\pi}{8}, \quad k = -4 \]

- In Matlab: \( n = 0 \ldots L - 1, k = 0 \ldots L - 1 \)
- fftshift(fft(f)) performs a shift by \( \frac{L}{2} \) (modulation of the signal by \( e^{-j2\pi L/2} = e^{-j\pi} = -1 \))

Derivative Filters (continued)

Two simple filters

Remember the problem we stated in the previous lecture:

- We are given a sampled signal and want to approximate the derivation of such a signal by “designing” a discrete filter \( h \).

- In the Fourier domain, \( H(\omega) = H(\omega + 2\pi) \) since the signal and filter are discrete.

- We want \( H(\omega) \) to be as close as possible to \( j\omega \), for \( \omega \in [-\pi, \pi] \)
Last time we found that the filter \( h[n] = [1 - 1] \) verifies:

\[
h[n] \xrightarrow{DFT} H(\omega), \quad |H(\omega)| = \left| 2 \sin \frac{\omega}{2} \right|
\]

A more symmetrical form would be:

\[
f[n] * h[n] = \frac{1}{2}(f[n + 1] + f[n - 1])
\]

(In this case we are trying to approximate \( \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) + f(x-\varepsilon)}{2\varepsilon} \))

Here \( h[n] = [1/2 \ 0 \ -1/2] \xrightarrow{DFT} \frac{1}{2} e^{-j\omega} - \frac{1}{2} e^{-j\omega} = j\sin(\omega) \)

The two approximations are represented in figure 1. Note that while the second filter we tried is a much better approximation of the first one, these two filters are a good approximation only for low frequencies.

Figure 1: The derivation and its approximation by two simple filters. We represented the magnitude of the spectrum.

### Ideal Filter

What would be the ideal derivative filter? Remember that we multiplied the spectrum with the box function to prove the sampling theorem, which corresponded to convolving the original signal with sinc \( \left( \frac{\pi}{T} \right) \) (where \( T \) is the sampling interval).

If the original signal is \( h[n] \), the reconstructed signal is the following (convolution with the sinc interpolation function): (best reconstruction according to Shannon’s theorem)

\[
h(t) = \sum_{n=0}^{L-1} h[n] \text{sinc} \left( \frac{\pi}{T} (t - n) \right)
\]
Note the above is now a continuous signal. Let’s compute its derivative:

\[
\frac{d}{dt} h(t) = \sum_n h[n]d(t-nT)
\]

where \( d(t) = \frac{d}{dt} \left( \frac{\sin(\frac{\pi t}{T})}{\frac{\pi t}{T}} \right) = \frac{1}{(\pi T)^2} \left[ \frac{\pi T}{T} \cos \frac{\pi t}{T} - \frac{\pi}{T} \sin \frac{\pi t}{T} \right] \)

What we get is an infinite signal discretized as follows (see figure 2)

\[
\ldots \quad 1/5 \quad -1/4 \quad 1/3 \quad -1/2 \quad 1 \quad 0 \quad -1 \quad 1/2 \quad -1/3 \quad 1/4 \quad -1/5 \quad \ldots
\]

Figure 2: Derivative of the sinc interpolation function.

If we compute the Fourier transform of this, we see that it approximates \( j\omega \) but is wiggly.

**Least squares filter design**

Given constraints on the length of my filter, how can we obtain the best derivative filter?

- The length should be \( L \), so there are \( L \) unknowns
- The target is \( j\omega \): more concretely we want \( \sum_{n=0}^{L-1} h[n]e^{-j\omega n} \approx j\omega \) for all \( \omega \)

It is not possible to solve this system for all \( \omega \), so we solve a least-squares problem:

\[
\min_{h[n]} \int_0^\infty \left( \sum_{n=0}^{L-1} h[n]e^{-j\omega n} - j\omega \right)^2 d\omega
\]

This solves the length constraint problem but does not solve the smoothing problem. For smoothing we apply the Fourier transform of a Gaussian:

\[
j\omega e^{-2\sigma^2 \omega^2}
\]
2D Fourier Transform

\[ f(x, y) \leftrightarrow \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)e^{-j\omega_x x + j\omega_y y} dxdy \]

\[ F(\omega_x, \omega_y) \leftrightarrow \left( \frac{1}{2\pi} \right) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(x, y)e^{-j\omega_x x + j\omega_y y} dxdy \]

Shift

\[ f(x-x_0, y-y_0) \leftrightarrow F(\omega_x, \omega_y)e^{-j(\omega_x x_0 + \omega_y y_0)} \]

The shift is not as obvious to obtain as in the 1D case. This can be a problem in motion estimation.

Affine transformation We note \( A \begin{bmatrix} x \\ y \end{bmatrix} \) a linear transformation of the 2D space.

\[ f(a_{11}x + a_{12}y, a_{21}x + a_{22}y) \]

\[ = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(A(x, y)) \exp \left(-j\begin{bmatrix} \omega_x & \omega_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}\right) dxdy \]

\[ = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x', y') \exp \left(-j\begin{bmatrix} \omega_x & \omega_y \end{bmatrix} (A^{-1}) \begin{bmatrix} x' \\ y' \end{bmatrix}\right) \det(A^{-1}) dx'dy' \]

\[ = \det(A^{-1}) F \left(A^{-T} \begin{bmatrix} w_x \\ w_y \end{bmatrix} \right) \]

If \( A \) is a rotation \( R (R^T R = I) \), also \( \det(R) = 1 \), the formula is even simpler:

\[ R \begin{bmatrix} x \\ y \end{bmatrix} \leftrightarrow \mathcal{F} \left(R \begin{bmatrix} w_x \\ w_y \end{bmatrix} \right) \]

Separable functions

\[ f(x, y) = f_1(x)f_2(y) \leftrightarrow \int_{-\infty}^{\infty} f_1(x) e^{-j\omega_x x} dx \int_{-\infty}^{\infty} f_2(y) e^{-j\omega_y y} dy = F_1(\omega_x)F_2(\omega_y) \]

Examples:

- \( f(x, y) = \cos(\omega_0 x) \) (see figure 3)

It is important to notice that \( f(x, y) = f(x)f(y) \) where \( f(y) = 1 \) is a constant function. Therefore the Fourier transform is the following:

\[ f(w, y) \leftrightarrow \frac{1}{2} \left( \delta(\omega_x - \omega_0) + \delta(\omega_x + \omega_0) \right) \delta(\omega_y) \]

Figure 3: Unidirectional cosine
• $f(x,y) = \cos(\omega_1 x) \cos(\omega_2 y)$ corresponds to four points in the Fourier domain.

• $\cos(\sqrt{2}x - \sqrt{2}y)$: unidirectional cosine rotated of $\pi/4$

(to be continued)