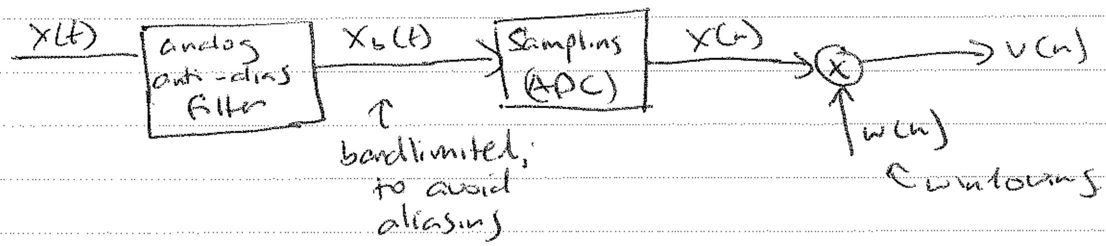
book sections:
P&M 7.4
w/ reference to
10.2

Class notes: Spectral analysis using DFT

1. motivation – see PPT slides

2. Basic processing flow and ideas

CT data is processed as follows



Once we have the sampled, windowed data v(n), we can take it's DTFT to find

$$V(\omega) = \underset{\cancel{DFT}}{\overset{DTFT}{}}\{w(n)\,x(n)\}$$

or $V(k) = DFT\{w(n)\,x(n)\}$

note that because our DFT will contain a finite # points, we always apply a window –
even if just a rectangular window to limit # pts.

~~Ignore this for now~~ - I used to teach filter design first. But, you will see windowing again!

recall windowing for FIR filter design:
→ we had a desired filter response $H_d(\omega)$
→ we used a window function to make a finite-length filter $h(n) = h_d(n)w(n)$
→ we got a frequency response that was a smoothed version of what we wanted : $H(\omega) = H_d(\omega) * W(\omega)$
→ we picked windows to trade off mainlobe / sidelobe

Here we will do something very similar
- → we'll apply windows to the data
- → our estimated spectrum will be a smoothed version of the true spectrum
- → the windows used, and tradeoffs, are basically the same.

<u>Example</u>: finding spectrum of a sinusoid using M-point boxcar window

our windowed signal is:

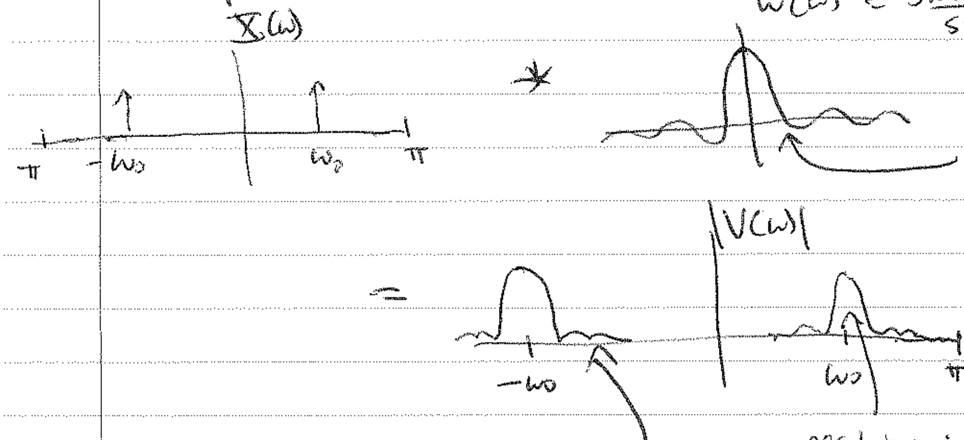
$$v(n) = x(n)\,w(n)$$
$$= \begin{cases} x(n), & 0 \le n \le M-1 \\ 0 & \text{else} \end{cases}$$

in the frequency domain,

$$V(\omega) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\lambda)\,W(\omega-\lambda)\,d\lambda = X(\omega) * W(\omega)$$

in pictures:

$$W(\omega) = \frac{\sin \omega M/2}{\sin \omega/2}$$



1st zero at $\frac{\omega M}{2} = \pi$

or $\omega = 2\pi/M$

so $ML = 4\pi/M$

resolution: ability to estimate ω_0 ~ mainlobe width

"leakage"
energy from the signal has "leaked" into other frequencies due to windowing

(3) Effect of window selection

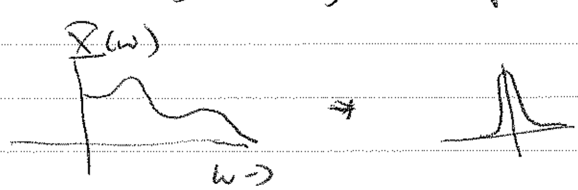
Just as with FIR filter design, we can trade off <u>mainlobe width</u> vs. <u>sidelobe levels</u>

see PPT & book examples

mainlobe / sidelobe tradeoffs found in Table 10.2

The window can have significant effects:

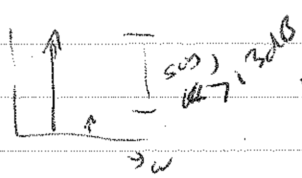
→ if mainlobe ~~#~~ width < spectral feature width, we get a good "picture" of the spectrum



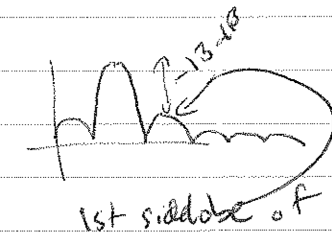
$= V(\omega) \approx X(\omega)$

but if mainlobe width is bigger, we mainly see shape of window (undesirable)

→ If sidelobes too high, ~~[illegible]~~ strong sources can mask weak ones



may look like, if we use boxcar →



1st sidelobe of strong src masks the weak source

4 Two common "illusions"

Illusion 1 undersampling in frequency may make it look as if spectrum is sparser than it is

"picket fence" effect

see book Fig 7.1.6 + related discussion for an example.

How to "fix"? If we zero-pad, we can get a denser frequency sampling + see the spectrum shape more clearly

→ Matlab examples: available on Trunk
picket_fence_~~demo~~ illusion.m
→ you will look at this in homework.

Illusion 2 zero-padding increases the true resolution of my spectrum estimate.

False! zero-padding gives a nicer display but isn't adding any new information.

The resolution is ~~set by~~ the window mainlobe width, which is set by the window type + length.

Matlab example: on Trunk
zero_pad_illusion.m

topic 5 Noise bandwidth

consider a sinusoid in noise

$$x(n) = A\sin\omega_0 n + \text{noise}$$

If noise is uncorrelated from one sample to another, its autocorrelation will look like

$$r_{xx}(l) = \sigma^2\delta(l)$$

which Fourier transforms to a flat spectrum — "white noise"

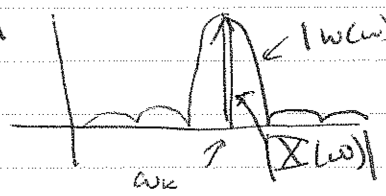
Typically we talk about AWGN:

Additive	(instead of multiplying signal)
White	(uncorrelated sample-to-sample)
Gaussian	(each sample comes from Gaussian distribution)
Noise	

~~We can th~~

When we do spectral analysis we get a combination of noise + signal in each frequency bin.

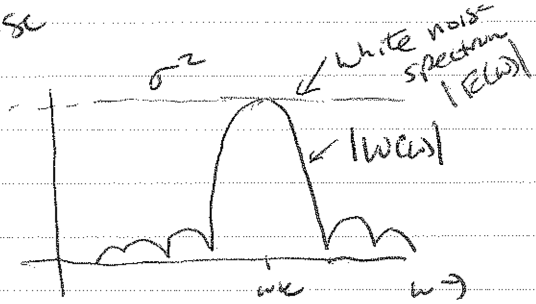
① signal



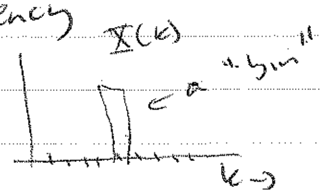
← picture for rectangular window

for a sinusoid perfectly centered in our bin, the signal is passed through

② noise



note: "bin" means the region of frequency around ~~and~~ each DFT/FFT center frequency



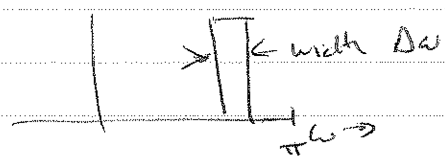
(OVER)

"noise" continued ..

From the last picture, we can see that the noise output for the k^{th} bin will contain noise from many frequencies

- sidelobe leakage
- mainlobe

If our windows gave rectangular windows in the frequency domain, it would be easy to calculate SNR in the bin



→ our signal would pass thru

→ noise would be attenuated by factor of $\frac{\Delta\omega}{\pi}$

so SNR would be

$$SNR = \frac{S}{\frac{\Delta\omega}{\pi}\sigma^2}$$

a longer window (bigger M) makes mainlobe smaller, so ~~M~~ $\Delta\omega \downarrow$ and SNR $\uparrow$

Real windows don't do this, but people have computed noise-equivalent bandwidths for various windows.

Can be used for comparison.

However, longer windows always help the SNR — so integration time is good for weak signals in noise

Matlab example: integration_time_example.m //