Administrative

- Exam 2 is on Wednesday
- Review session tonight, 7-8 pm, hopefully this room (bring specific questions)
- Matlab/Python 5 is due Thursday at midnight (so can use Thursday afternoon office hours)

Exam 2 coverage

- Exam covers material from lectures 10-16 (+ spectrogram examples at start of Lecture 17), and associated HW (including today's assignment!)
  - Multi-taper analysis from Lecture 17 is not on any exams
- Spectrum estimation material is mainly off course notes, not really related to book
  - The book does cover periodograms, but in more detail than the course notes did. If you understand notes and HW, should be OK
- Could ask questions on compression related to Matlab/Python project 4; will **not** assume you've done #5 (but as mentioned before, wouldn't hurt to at least read it)

- EASY POINTS: Expect a question on the following three slides from FFT lecture, addressing question: besides FFT, what other transforms might you want to use?

Coverage of prior material..

- While exam won't explicitly cover prior material, we have been using results from the first part of the course. These include
  - Common Fourier transform pairs (sinc <-> boxcar, cosines, sines, etc.
    - Haven't really used pulse train recently – that is mostly needed for sampling theory
  - Fourier transform properties (conjugate symmetry, multiplication <-> convolution, etc)
- Don't need to review prior lectures, but you may want to include some of the above on your cheat sheet

(Review me) Related transforms - Discrete Cosine Transform

- Uses Fourier transform property: if x(n) is real and even, X(k) is real and even.
  - For real and even x(n), exponential in DFT can be replaced by a cosine, since sine only contributes to imag(X(k)).
- To use the DCT, we copy (*extend*) non-symmetric signals to make them symmetric: see below.

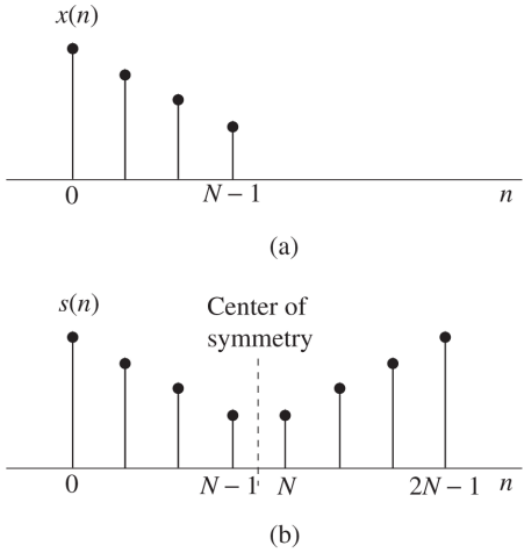


Figure 7.5.1 Original sequence $x(n)$, $0 \le n \le N-1$ and its $2N$-point even extension $s(n)$, $0 \le n \le 2N-1$.

(Review me) Why use the DCT?

- Compared to the DFT, the DCT gives a less compact representation of sinusoids (Figure 7.5.2, book), but more compact representation of other signals . Here, 'compact' means 'fewer coefficients with significant amplitude'
- DCT turns out to be good for many images
- DCT is the basis of the JPEG image compression algorithm

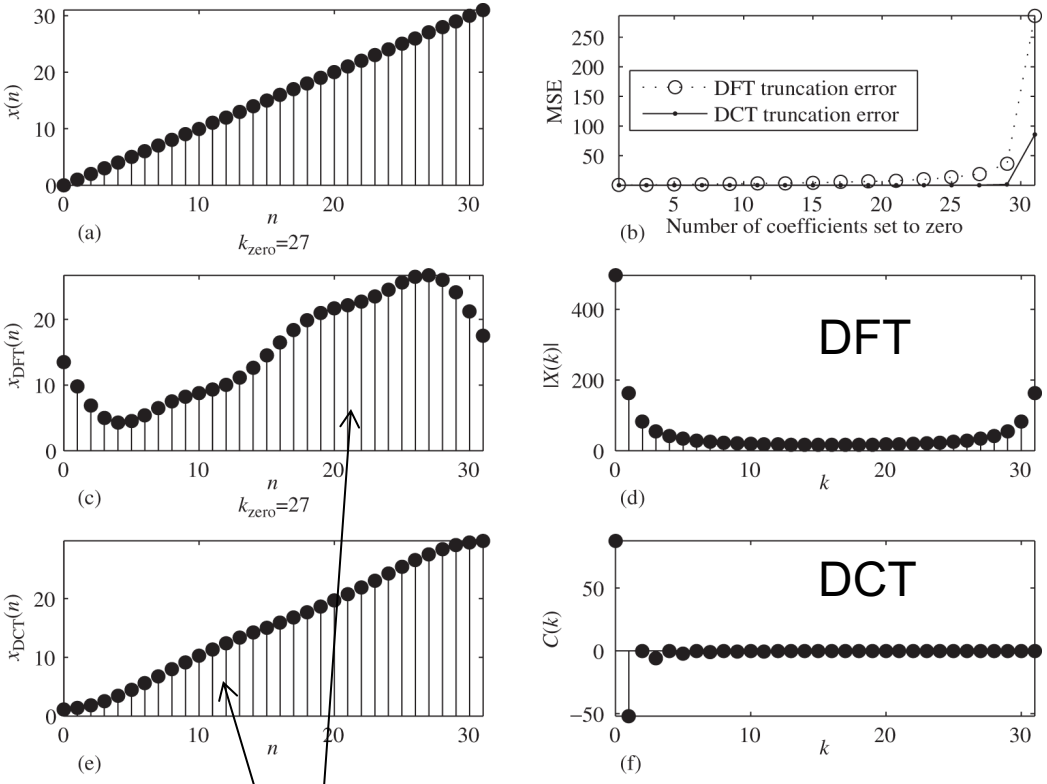


Figure 7.5.3 A discrete time sinusoidal signal and its DFT and DCT representations.

Reconstructions using 3 coefficients

Tufts

(Review me) Other algorithms to know about

- Goertzel (P&M 8.3)
  - Rewrite the DFT so we have a parallel bank of filters, each one of which gives the output for a single frequency in the DFT
  - Advantage is that we don't need to implement every frequency; so can be faster than FFT if we just need answers at a few frequencies
  - Classic use: processing of dial tones
- Chirp z-transform (P&M 8.3)
  - Lets us evaluate the transform at points other than the unit circle
  - Used in speech analysis (on-line, see "The Chirp z-Transform Algorithm—A Lesson in Serendipity")

EE-125:
Digital Signal Processing

FIR Filter Design 2

Professor Tracey

Tufts

"Big picture"

- Basic idea of FIR filter design:
  - Decide on the ideal frequency response $H(\omega)$
  - Pick a form for the filter that guarantees linear phase FIR
  - Find the coefficient weights that best match $H(\omega)$
- Last step usually involves trading off fast frequency transitions to get lower responses elsewhere
- Three main design approaches:
  - Window method (matlab 'fir1' command)
  - Frequency sampling (matlab 'fir2' command)
  - Optimization of a cost function (ex: 'firls' or 'firpm')

Outline

- Filter design by windowing
  - Review last time
  - Talk about Kaiser window – the super-window
- Filter design by frequency sampling
  - Simple versions
  - More optimal

Filter response specifications

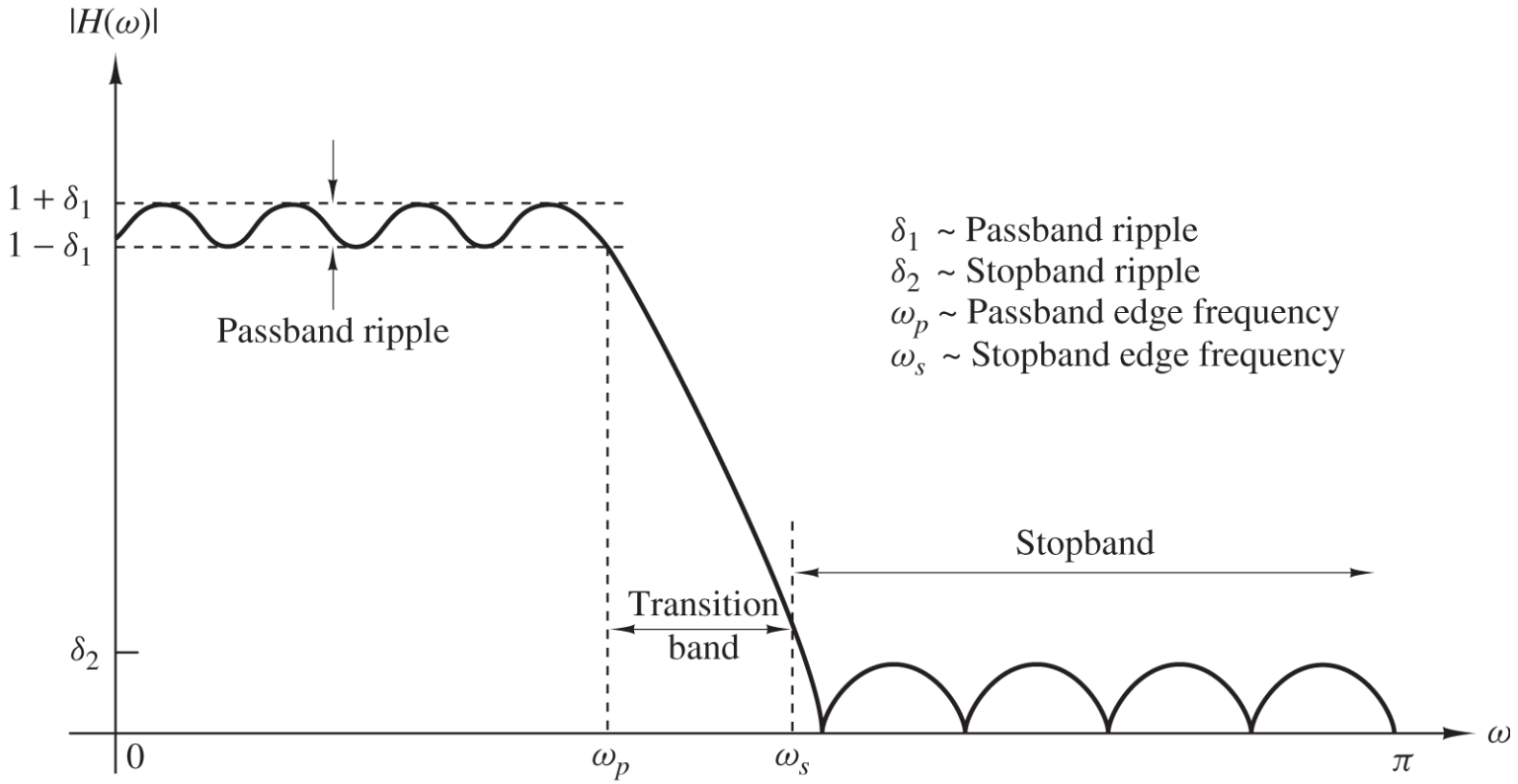
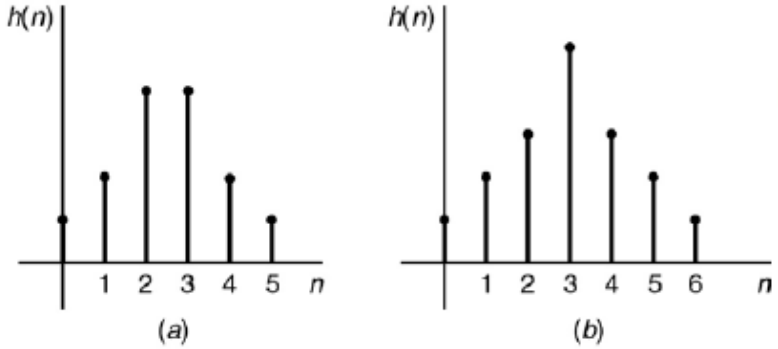


Figure 10.1.2 Magnitude characteristics of physically realizable filters.

Tufts

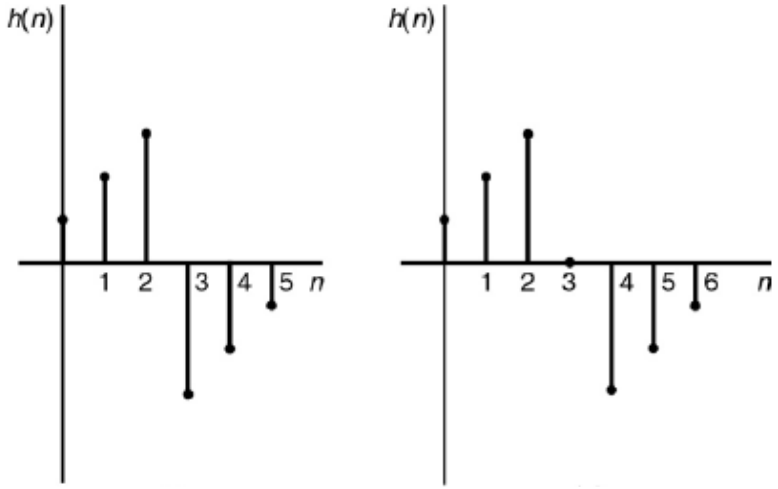
Four basic FIR forms that give linear or generalized linear phase

FIR II: even length, symmetric



FIR I: odd length, symmetric

FIR IV: even length, antisymmetric



FIR III: odd length, antisymmetric

Note for this case that h[M/2]=0

Tufts

Window design considerations -1

- The main issues are main lobe width and sidelobe height
- Main lobe determine how 'smoothed' the desired response is
  - Thus ideally, main lobe would be very narrow
  - This generally implies higher M
- The sidelobe determines 'ringing' in frequency
  - High sidelobes mean less attenuation of undesired frequencies
  - High frequency sidelobes result from sharp discontinuities in time

- Rectangular window: narrowest mainlobe, but high sidelobes
- Other windows; wider mainlobe but lower sidelobes
  - All are linear phase
  - All have smaller mainlobe as M increases (just like rect)
  - Design is generally trial-and-error

Some details…

- See examples filteringExamplesWindowing.m and kaiserDemo.m

- Look at ‘fir1’ documentation – note how frequencies are specified (Fs/2 Hz = pi radians is called ‘1’)

Window design considerations -2

- Effect of window length
  - As the window gets longer, the mainlobe gets narrower (think of rect <-> sinc)
  - A conservative rule of thumb is that the filter's transition from mainlobe to sidelobe (in frequency) is ~same range as mainlobe width in frequency
  - Window length also determines where sidelobes start
  - Longer windows mean more computation
- Tradeoffs in window type
  - Do we want a window whose response drops off quickly but then remains flat (Hamming), or one whose response drops off more smoothly but continuously (Hann?)
  - Depends partly on application – do we want to remove a single interfering source, or noise across all bands?

Questions

What might be reasonable window types for FIR filter design in each of these cases?

1. The signal of interest ‘x’ is at 90 Hz. There may be a second signal that could be extremely close in frequency
2. You are processing EEG recordings for biofeedback, with the max frequency of interest being 50 Hz. There is noise at all frequencies.
3. You are processing the same EEG recordings as in #2, but realized there may be power line interference at 60 Hz.

Outline

- Filter design by windowing
  - Review last time
  - Talk about Kaiser window – the super-window

- Filter design by frequency sampling
  - The basic idea
  - A very simple version: take an FFT, zero out the "bad" frequencies, take IFFT
    - let's analyze this…. Not ideal
  - Example of how a transition band helps us

FIR design by frequency sampling

- Basic idea
  - Specify desired response at evenly spaced samples in frequency (why evenly spaced?)
  - Make sure Hd(ω) is conjugate symmetric (why?)
  - Make sure Hd(ω) is has linear phase (what slope?)
  - Take IDFT / IFFT to find h(n)
- Variations
  - Use basic idea, but use large # points in frequency; then window / truncate the calculated h(n) (matlab FIR2)
  - Basic idea, but at unevenly spaced points and solve matrix eqn
  - Specialized formulas for special cases (P&M 10.2.3)
- Pluses: approach allows arbitrary filters; Minus: need to beware of oscillation between points specified

Next topic…..

- Filter design by least squares

- A first approach which has is “optimal” in some way
  - Approaches so far have been basically ad-hoc (FIR design by window, etc)

Optimal FIR filter design

- Given all the values for stopband, pass band, allowed ripple, desired attenuation, how do we **optimally** design an FIR filter?

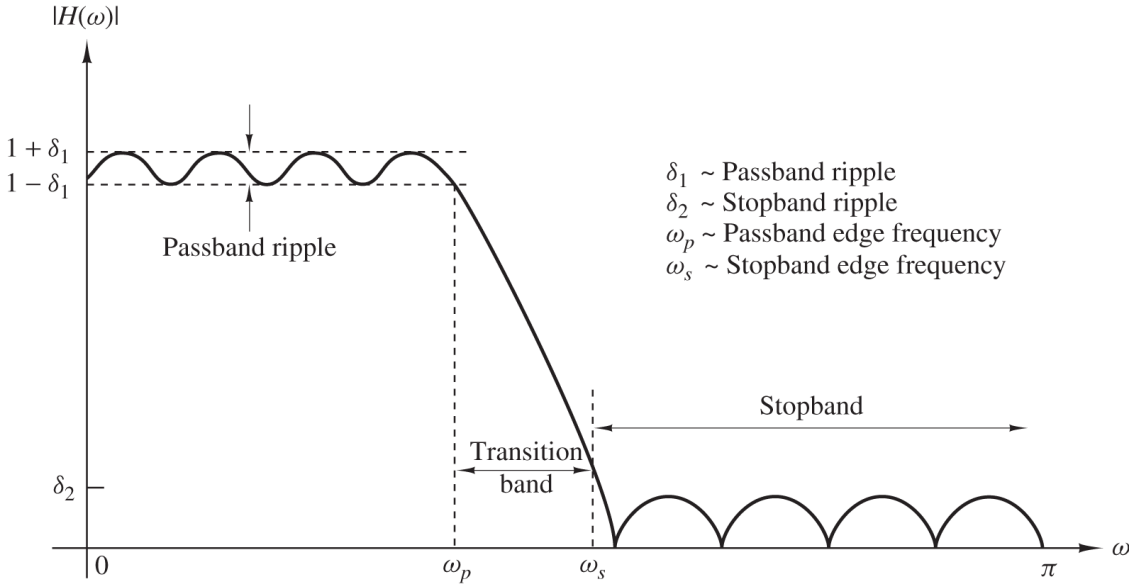


Figure 10.1.2 Magnitude characteristics of physically realizable filters.

- This discussion isn't from book: instead see:

https://ccrma.stanford.edu/~jos/sasp/Optimal_FIR_Digital_Filter.html

Notation: Lp norms

- If x is a vector, it's L-p norm is defined as

$$\|\vec{x}\|_p = \left(\sum_{i=0}^{N-1} |x_i|^p \right)^{1/p}$$

- Important cases:
  - p=0; number of non-zero elements in x
  - p=1; "city block distance", sum of absolute values
  - p=2; Euclidian distance (Pythagorus)
  - p = infinity; max value of x
- Example: if x = [1,4,0], then
  - ||x||_0 = 2
  - ||x||_1 = 1+4 +0= 5
  - ||x||_2 = sqrt(1^2 + 4^2+ 0^2) = 4.12
  - ||x||_20 = 4.00000000000001
  - ||x||_infinity = 4

FIR filter design problem becomes:

- Pick a filter design *H* that minimizes

$$\|W(\omega_k)|H(\omega_k) - D(\omega_k)|\|_p$$

where ω_k are frequencies where response is specified, *H* is the filter we are designing, *D* is the desired response, and *W* is an optional weighting

- We can optimize the least-squares criterion by setting p=2; minimize the squared, summed differences between *H* and *D*
- How do we write *H* in terms of our FIR filter coefficients h(n)?

Filter design matrix

- Make some assumptions:
  - Filter is of length L+1, where L is an even number
  - Filter is centered around n=0 (we can shift it later to make it causal)
  - We want a linear phase filter, so make it symmetric
- Then, we can find $H(\omega_k)$ by taking the DFT of h(n):

$$H(\omega_k) = \sum_{n=-L/2}^{L/2} h(n)e^{-j\omega_k n}$$

$$= h_0 + 2\sum_{n=1}^{L/2} h(n)cos(\omega_k n)$$

- We can collect these into a big matrix, one row for each k
- Typically, # frequencies k > filter length L+1

Least-squared filter design continued

- We can collect the matrix into A, the filter coefficients into a vector *h*, and the desired response into vector *d* (*)

- Now, our problem in matrix form is:

$$\min_h \|A\vec{h} - \vec{d}\|_2$$

- The solution is found by expanding out the matrix-vector terms, taking 1st derivative, and setting it to zero (*)

$$\hat{\vec{h}} = [(A^T A)^{-1}]A^T \vec{d}$$

* (details in handwritten notes)