SPRING 2023

CS 378: INTRO TO SPEECH AND AUDIO PROCESSING

Digital Signal Processing for Speech 2

DAVID HARWATH
Assistant Professor, UTCS

The University of Texas at Austin
Department of Computer Science
College of Natural Sciences

Why we can't compute the DTFT

- The DTFT produces a *continuous valued spectrum by taking an infinite sum of a discrete time signal*
  - We still can't compute the DTFT with a digital computer

- We need to further modify the DTFT to take a finite, discrete-time signal as input and output a finite, discrete-frequency spectrum

Solution: Truncation

Let $w[n]$ be a length N window function defined as

$$w[n] = \begin{cases} 1: 0 \leq n \leq N-1 \\ 0: \text{otherwise.} \end{cases}$$

The product $\mathrm{x}[n]w[n]$ will be a length N signal → the DTFT of $\mathrm{x}[n]w[n]$ will no longer have an infinite summation.

The DTFT of a windowed function

$$\text{DTFT}(x[n]w[n]) = \sum_{n=-\infty}^{\infty} x[n]w[n]e^{-j\omega n} = \sum_{n=0}^{N} x[n]e^{-j\omega n}$$

The above function is a finite sum that we can actually compute, which we are free to query at any value of ω

Big caveat: computing the above function does not actually tell us what the DTFT of $x[n]$ is evaluated at ω. It tells us the DTFT of $x[n]w[n]$ evaluated at ω.

The DTFT of a windowed function

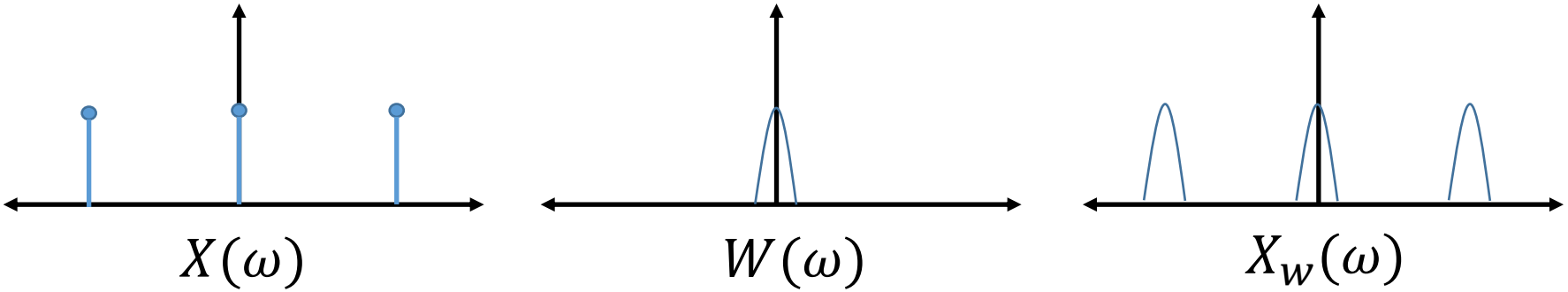
Windowing is multiplication in the time domain:

$$x_w[n] = x[n]w[n]$$

Which corresponds to *periodic convolution* in the *digital frequency* domain:

$$X_w(\omega) = X(\omega) \circledast W(\omega)$$

Therefore, windowing "smears" the spectrum $X(\omega)$



Discrete Fourier Transform (DFT)

It is most common to sample the the DTFT of the windowed signal $x[n]w[n]$ at N uniformly spaced frequencies between 0 and 2π (remember, one period has all the information)

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi k}{N}n}$$

- This is known as the **Discrete Fourier Transform**, and is what we will actually be computing in this class

Inverse DFT

The DFT is also an invertible transform. We can recover $x[n]$ from $X[k]$ using the Inverse DFT (IDFT):

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, e^{j\frac{2\pi k}{N}n}$$

Computing the DFT

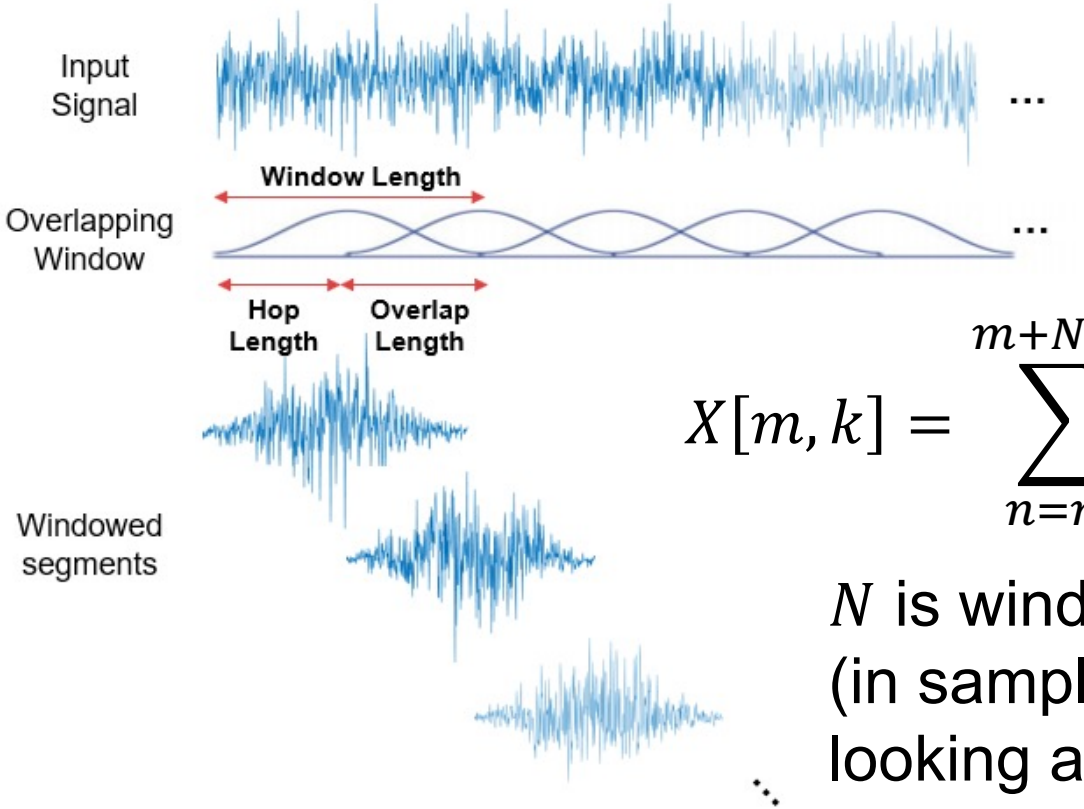
- The naïve implementation of the DFT sum is $O(N^2)$
- The *Fast Fourier Transform (*FFT) is an *algorithm* (actually, a family of algorithms) for computing the DFT in $O(N\ logN)$ time
- We won't have time to go into the details of how the FFT works, but I'll post some supplementary readings on Canvas if you are interested.

Zero Padding

If we want to compute more than N frequencies with the DFT, all we have to do is *zero pad* the input signal (**after** applying the window) to length $M > N$ and then compute the DFT of the zero padded signal

This can give us additional frequency resolution, **but our resolution is always fundamentally limited by the distortion introduced by the windowing function.**

Discrete Time STFT



When we have a long signal $x[n]$, we can choose a window function that is much shorter than $x[n]$ but make it *movable*

$$X[m,k] = \sum_{n=m}^{m+N-1} w[n-m]x[n]e^{-j\frac{2\pi k}{N}n}$$

N is window length, m is time shift (in samples) of the window we are looking at

Window Tradeoffs

There are 2 takeaways to remember when it comes to windows:

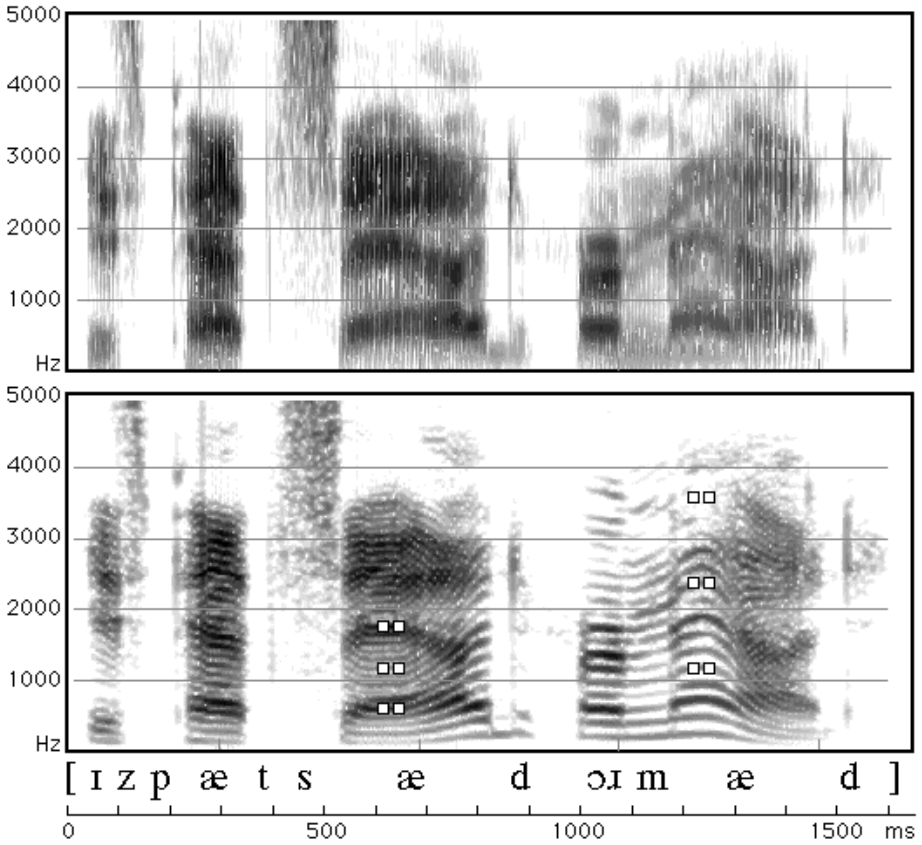
1. Window length controls the tradeoff between time resolution and frequency resolution. Larger window gives better frequency resolution, worse time resolution.

2. Window shape (in time domain) controls the shape of the distortion in the frequency domain. The tradeoff here is generally between main lobe width and side lobe energy.

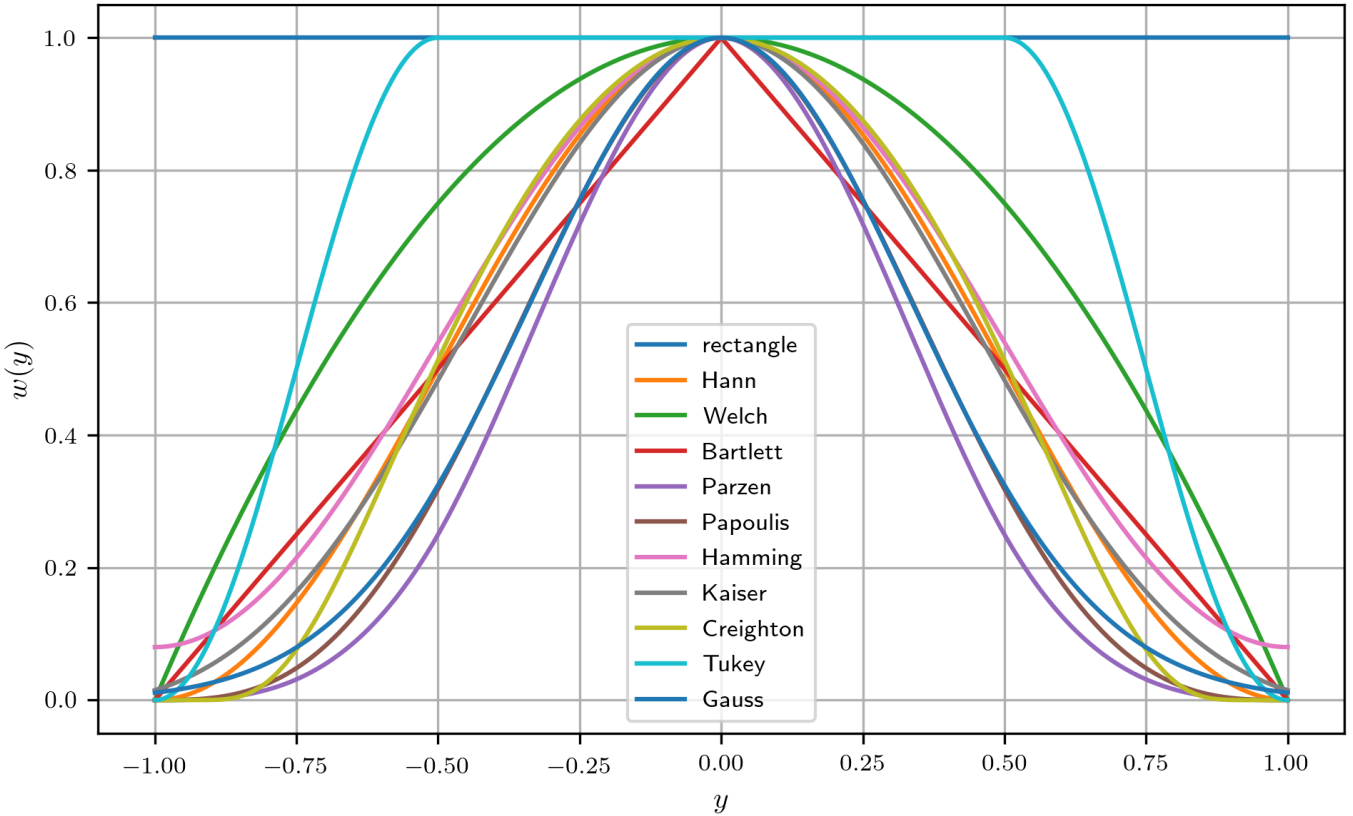
Comparison of Window Lengths

Short window (~5ms)
Better time resolution,
worse frequency resolution

Long window (~25ms)
Better frequency resolution,
worse time resolution



Window Shapes (Time Domain)

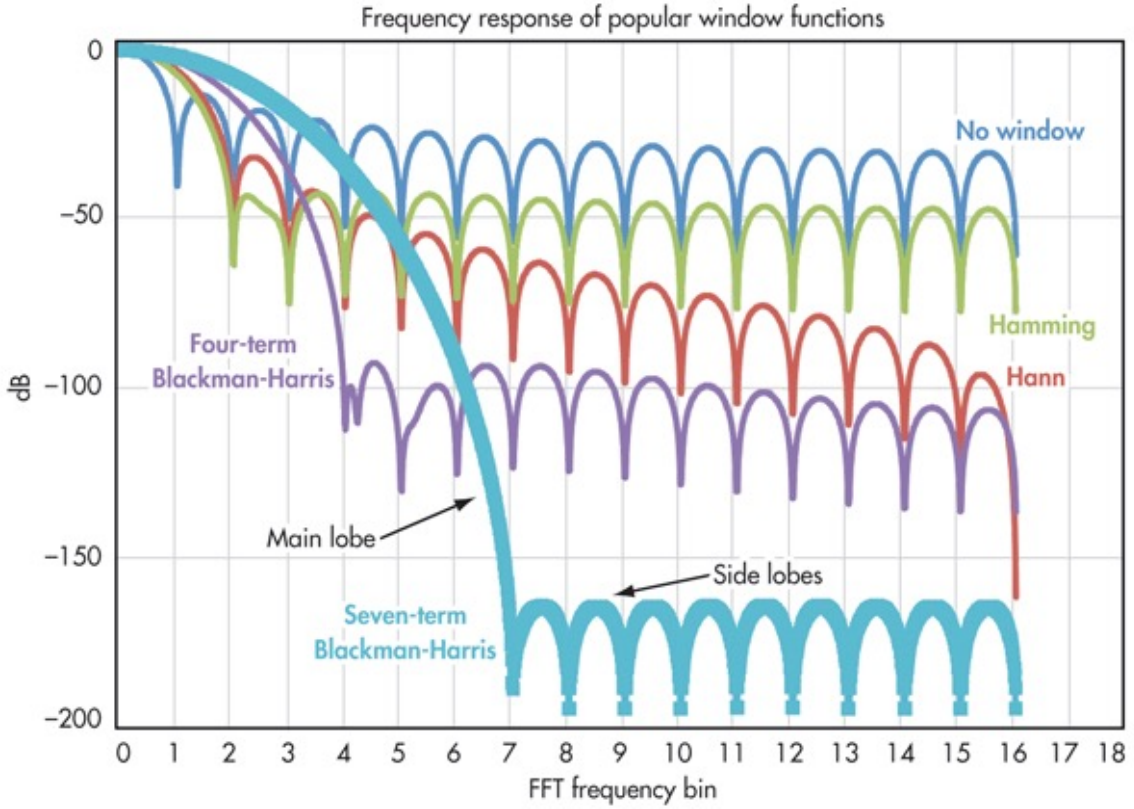


Window Shapes (Frequency Domain)

Main tradeoff: main lobe width vs. side lobe energy

Wide main lobe gives worse frequency resolution

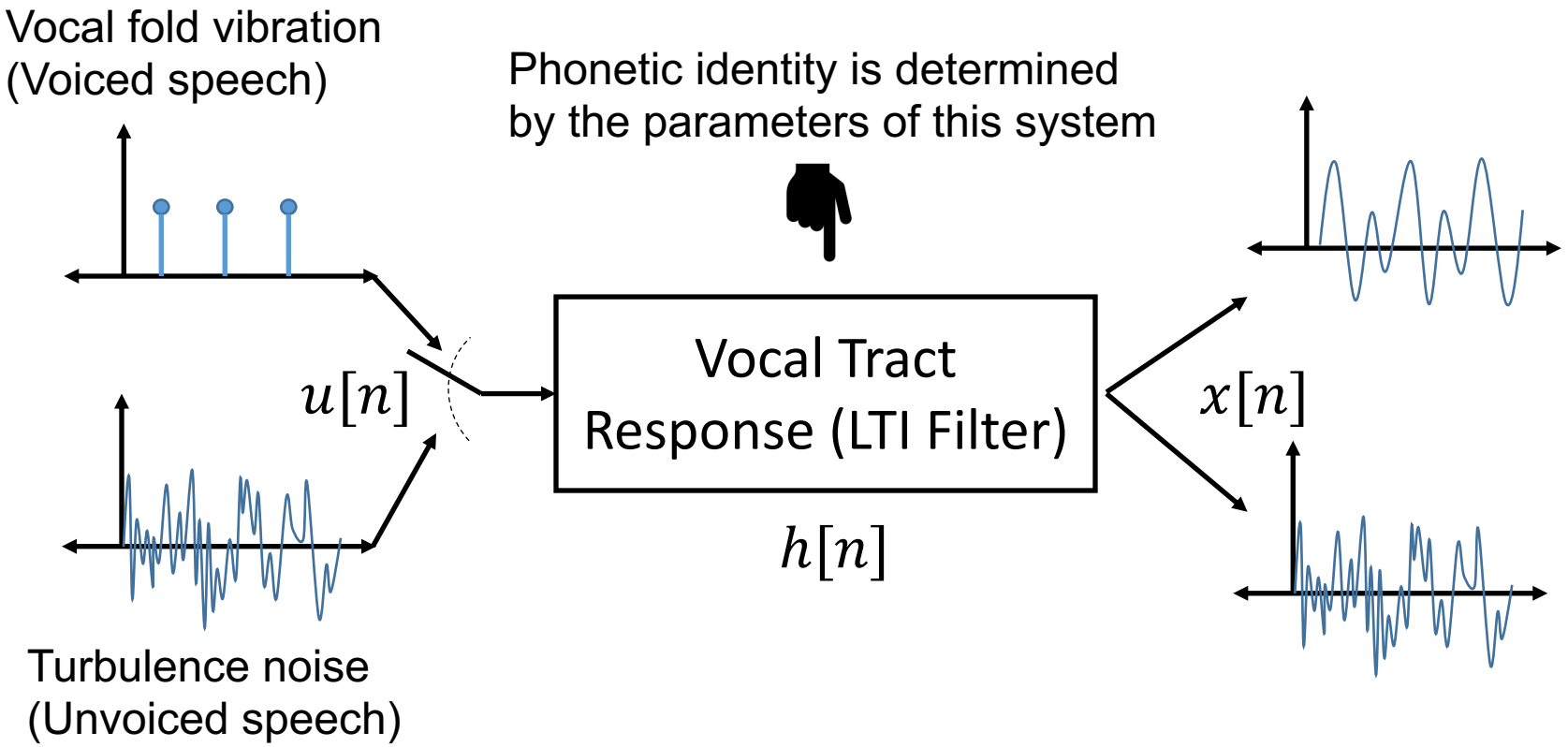
High side lobe energy can hide small peaks



Recall: Source-Filter model of speech

- Spectrum gives us the convolution of source and filter
- Phonetic identity is almost entirely reflected by the filter response
  - But we don’t know the source a priori
    - ”Blind” deconvolution to recover filter response and thus phonetic identity at a particular point in time

Recall: Source-Filter model of speech



Blind Deconvolution via Cepstrum

- We have that $x[n] = u[n] * h[n]$, and $X(\omega) = U(\omega)H(\omega)$
- If we take the log of $X(\omega)$, we get

$$\log\{X(\omega)\} = \log\{U(\omega)\} + \log\{\mathrm{H}(\omega)\} = \hat{X}(\omega)$$

- By the linearity of the IDTFT, we have that

$$\hat{x}[\mathrm{n}] = \hat{u}[n] + \hat{h}[n]$$

↑

"Cepstrum" of $x[n]$

Computing the Cepstrum

Note: $\log\{\mathrm{X}(\omega)\} = \log|\mathrm{X}(\omega)| + j\arg\{\mathrm{X}(\omega)\}$

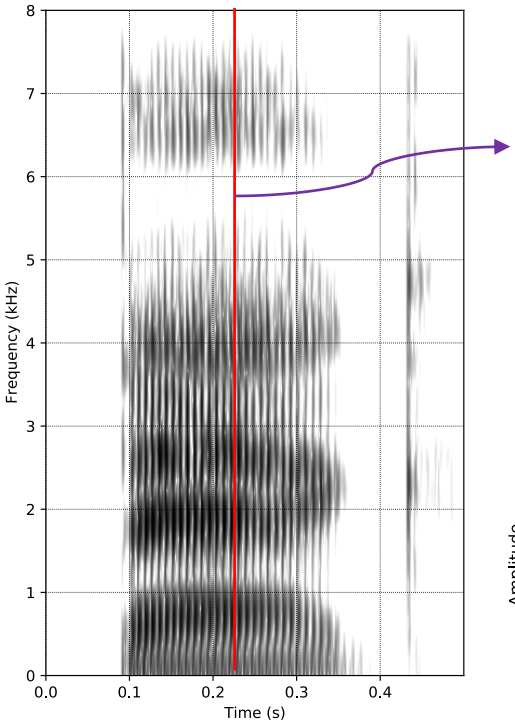
Complex Cepstrum:

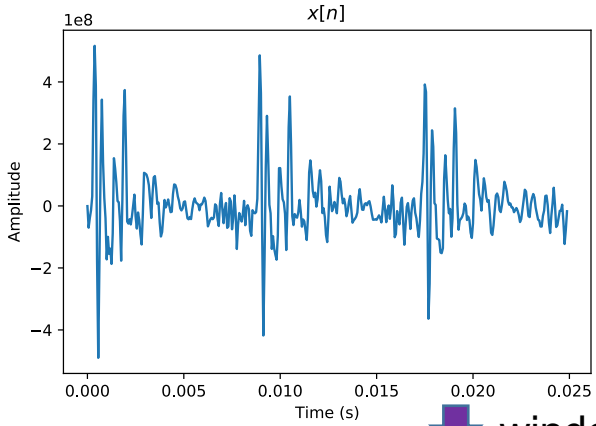
$$\hat{x}[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log\{X(\omega)\}e^{j\omega n}d\omega$$

(Real) Cepstrum:

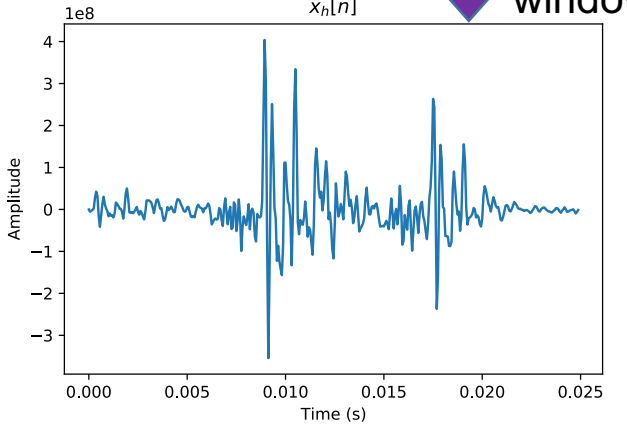
$$c[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log|X(\omega)|\, e^{j\omega n}d\omega$$

Cepstrum Example

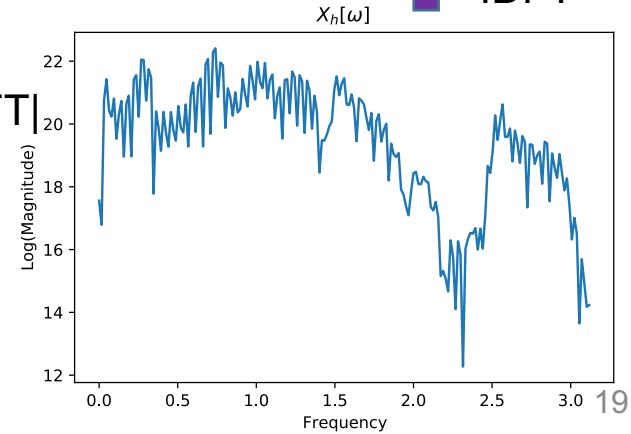




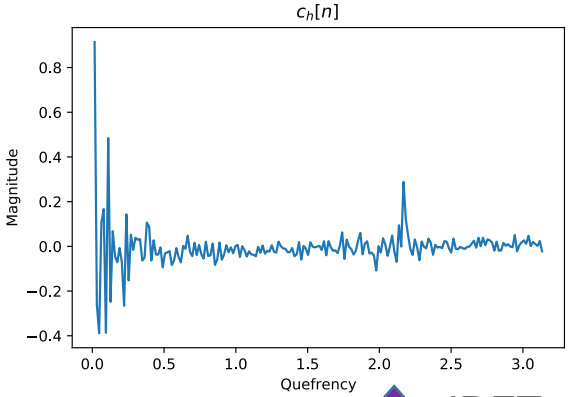
window



log|DFT|



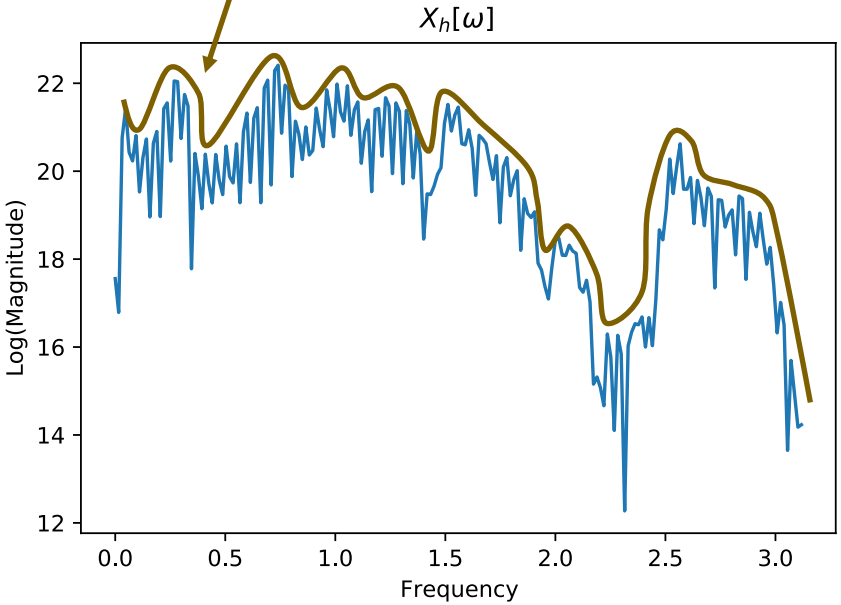
IDFT



Cepstrum Example

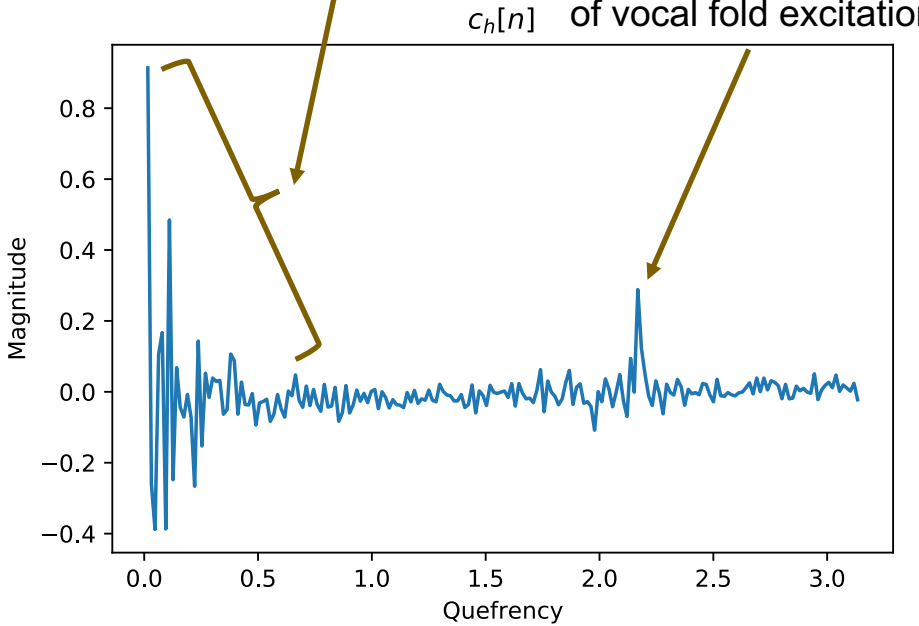
"Low frequency" envelope reflects vocal tract resonances (formants)

"High frequency" squiggles are harmonics of fundamental frequency of excitation (vocal fold vibration)



Low quefrency components capture shape of the spectral envelope

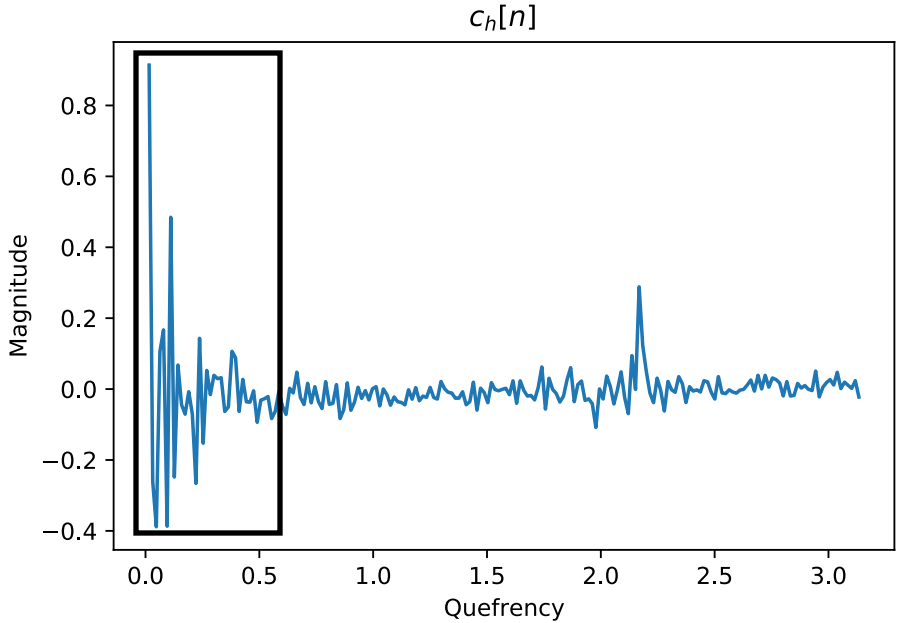
Horizontal location of this lone high quefrency peak indicates fundamental frequency of vocal fold excitation



Cepstral Liftering

We “lifter” the cepstrum by only keeping the first C components, discarding the rest. Generally, $C = 13$

This effectively removes the excitation information while keeping vocal tract resonance information, achieving the blind deconvolution we were aiming for.

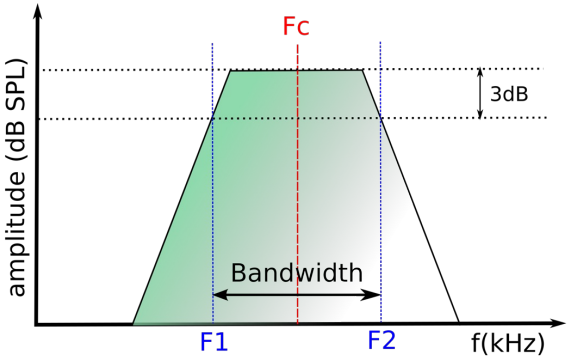


Inspiration from Psychoacoustics

There are several well-known results from psychoacoustics that we can take advantage of:

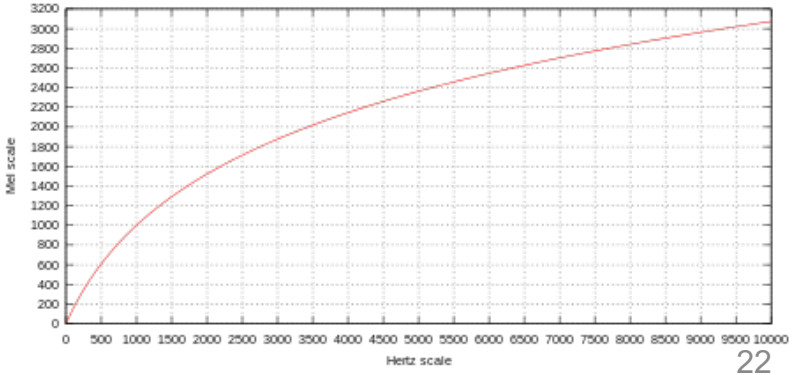
Critical Bands

Humans have trouble perceiving multiple tones that are very close together (frequency masking)

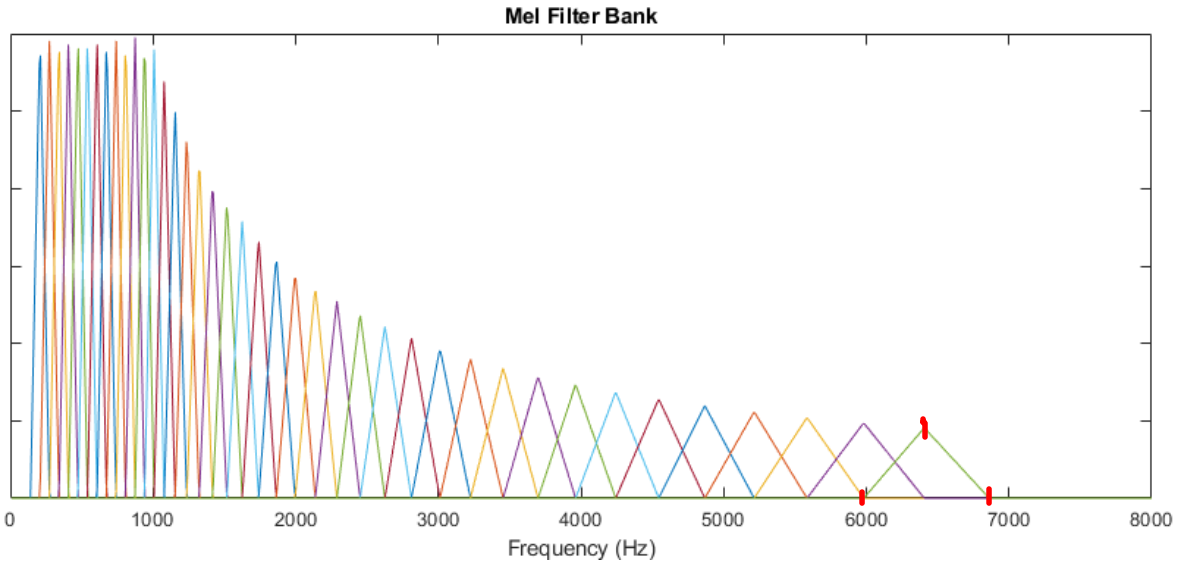


Logarithmic Pitch Perception

Human perception of pitch is generally logarithmic (connection to music: increase 1 octave = doubling of frequency)



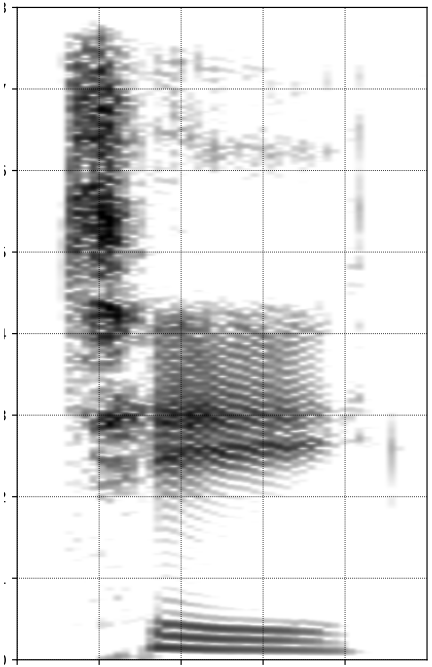
Mel Filter Banks



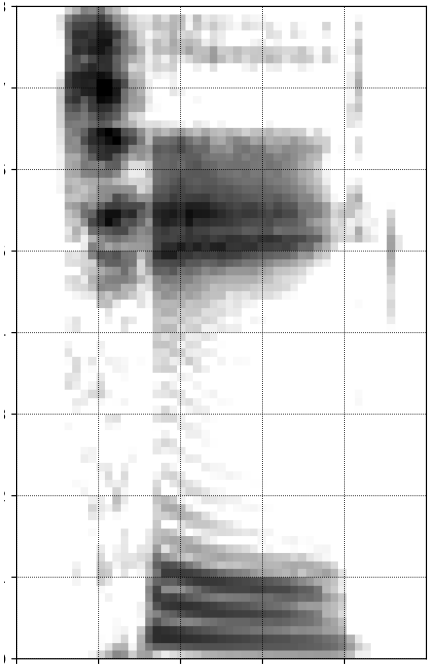
Mel filter banks combine the ideas of critical bands and logarithmic pitch perception.

Instead of directly using the magnitude/energy spectrum, we can sum up the energies underneath each triangular filter

mel Spectrograms



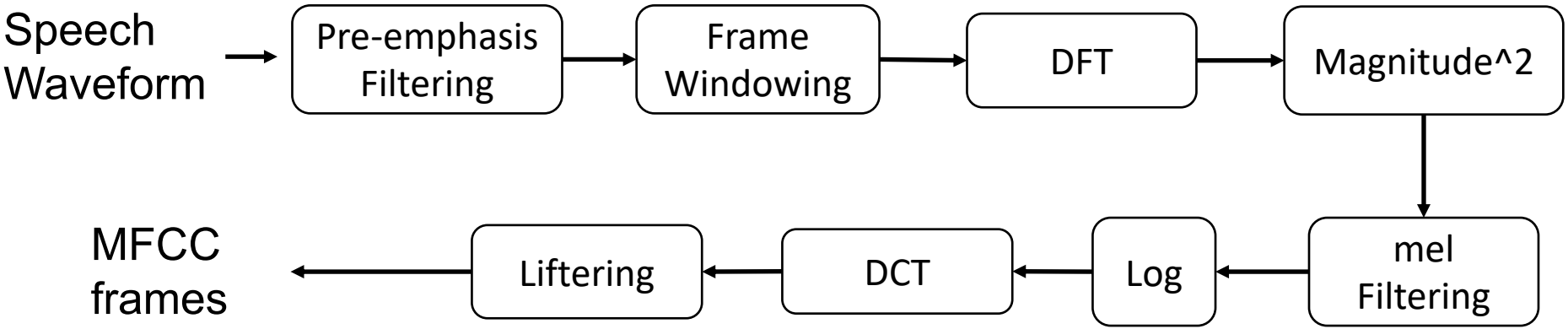
Linear (wideband)



Mel (wideband)

(Drumroll…) The MFCC Pipeline

Mel Frequency Cepstral Coefficients (MFCCs) are a very (the most?) popular feature extraction method for speech recognition.



IDFT vs. DCT

One little twist: practical MFCC implementations use the discrete cosine transform (DCT) instead of IDFT:

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

But wait, there's more!

- It is common to also estimate the first and second order time derivatives of the MFCCs and concatenate those to the MFCC vectors
- 13 MFCCs + 13 first derivatives (Δ) + 13 second derivatives (Δ Δ) = 39-dimensional features (MFCC39)
- Δ and Δ Δ features are generally computed dimension-wise via temporal differencing across consecutive frames

Current Trends

- Log mel filterbanks
  - DNNs handle correlated features well
  - Use lots of mel-frequency channels (40 – 120)
  - Stack lots of adjacent frames (for example, between 11-17)

- CPC, APC, wav2vec2.0, HuBERT
  - Self-supervised models that do some form of “language modeling” on speech
  - Think of it as trying to create a BERT for speech waveforms

- Raw waveform embeddings
  - Feed the raw waveform samples to a DNN!
  - Computationally expensive (high sample rate!)
  - Actually works for some applications