Speech Analysis and Feature Extraction

TTT4185 Machine Learning for Signal Processing

Giampiero Salvi

Department of Electronic Systems
NTNU

HT2020

Outline

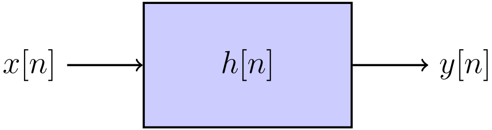
1. Speech Signal Representations
   - Signal Processing Reminder
   - Sampling and Quantization
   - Pre-Emphasis
   - Windowing
   - Discrete Fourier Transform
2. Feature Extraction
   - Linear Prediction Analysis (LPA)
   - Cepstrum
   - Perceptually Motivated Features

Assuming that you know

$$x[n] \longrightarrow \boxed{T\{.\}} \longrightarrow y[n]$$

- linear time-invariant systems (continuous and discrete time case)
- convolution
- impulse response
- Fourier transform and transfer function

Convolution and Impulse Response



$$y[n] = T\{x[n]\} = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n]$$

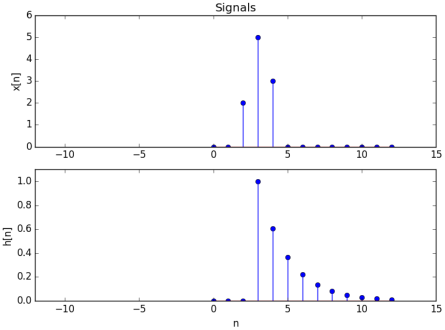
Properties:

- linearity: $(a_1 x_1 + a_2 x_2) * h = a_1(x_1 * h) + a_2(x_2 * h)$
- simmetry: $x * h = h * x$

Kind of complicated to interpret.

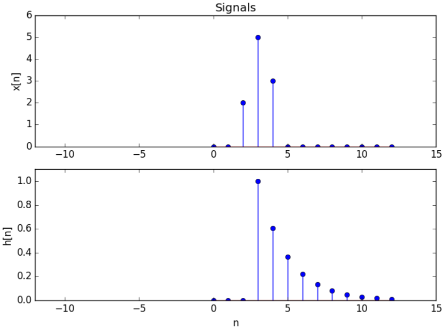
Convolution: Illustration

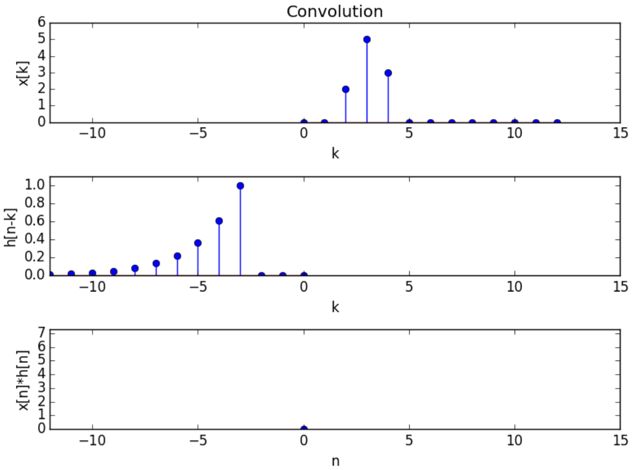
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



Convolution: Illustration

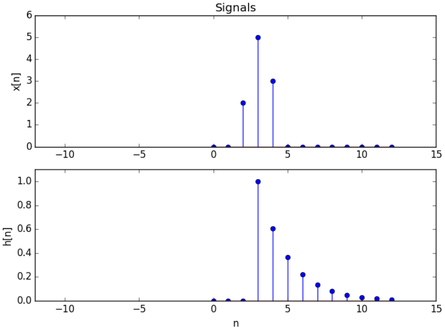
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

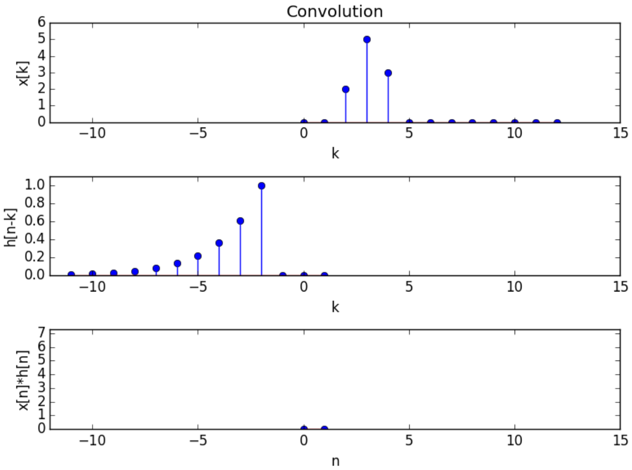




Convolution: Illustration

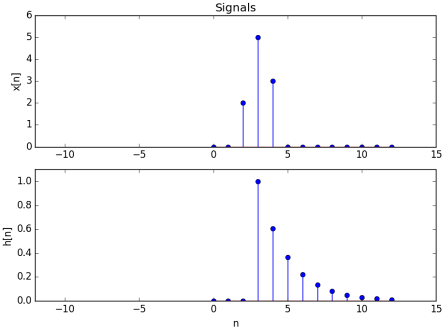
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

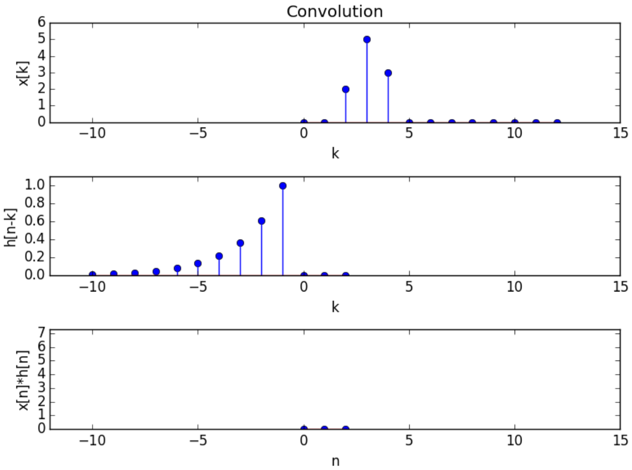




Convolution: Illustration

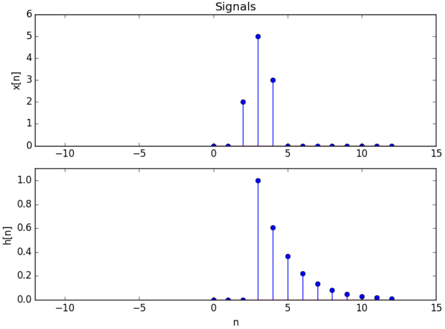
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

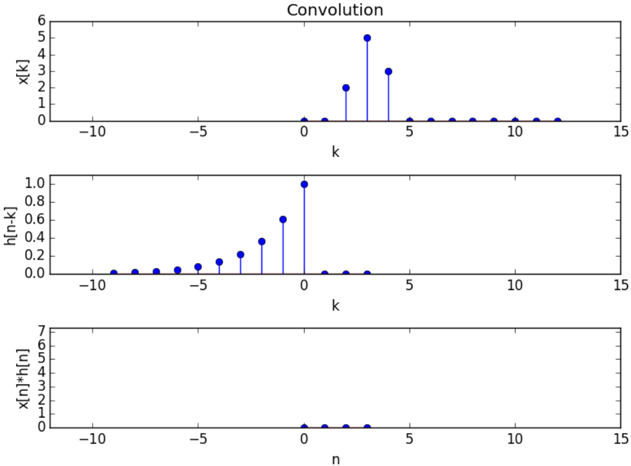




Convolution: Illustration

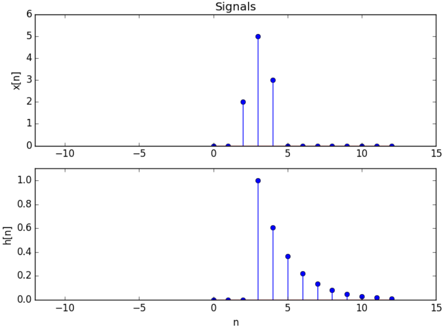
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

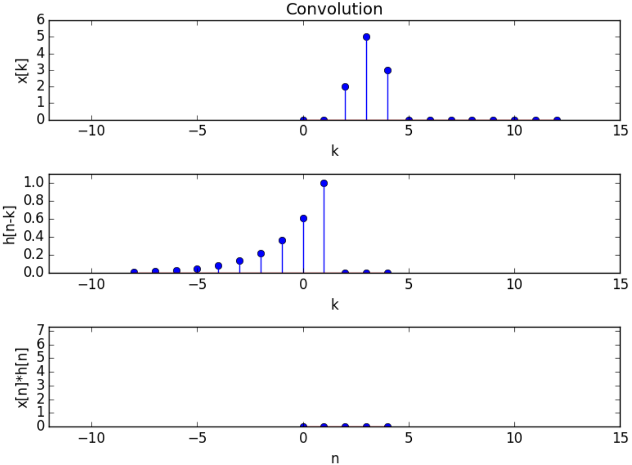




Convolution: Illustration

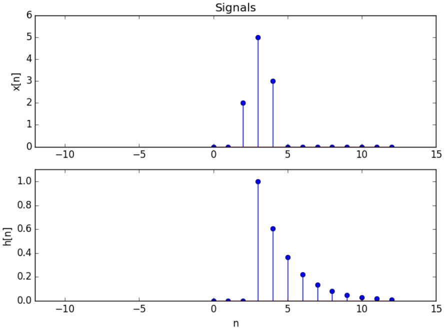
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

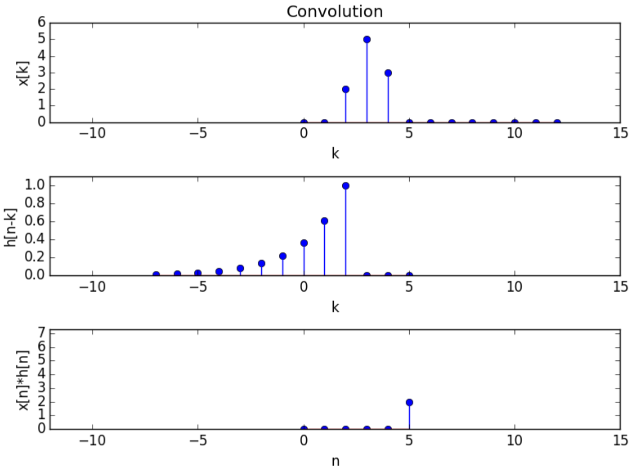




Convolution: Illustration

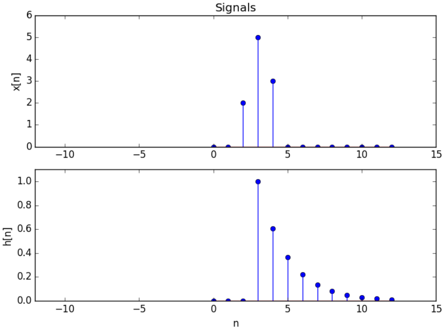
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

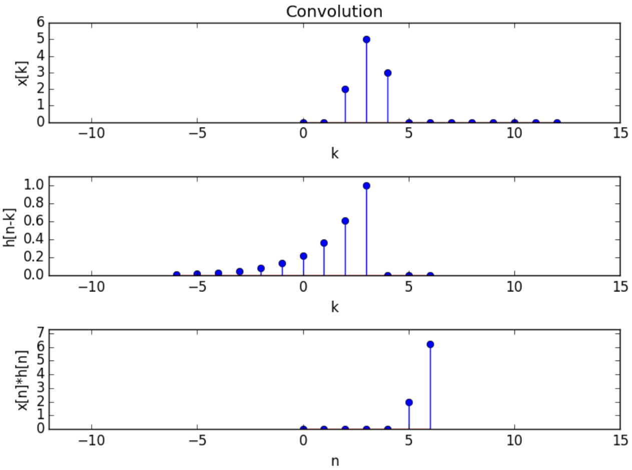




Convolution: Illustration

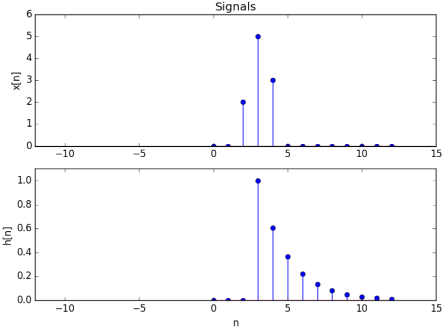
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

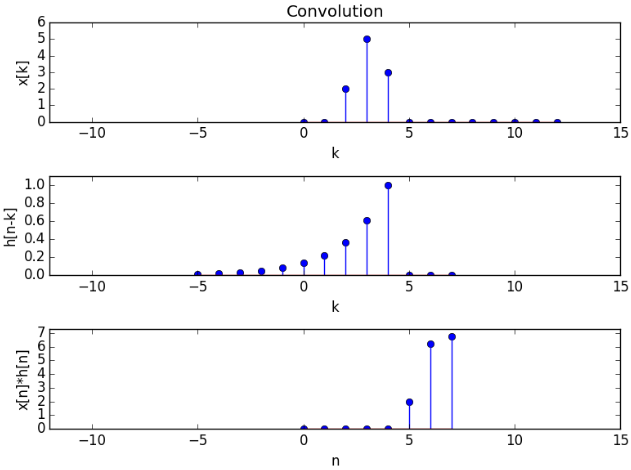




Convolution: Illustration

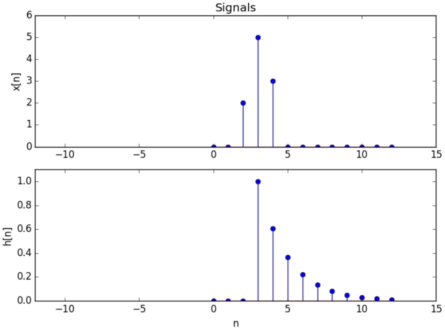
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

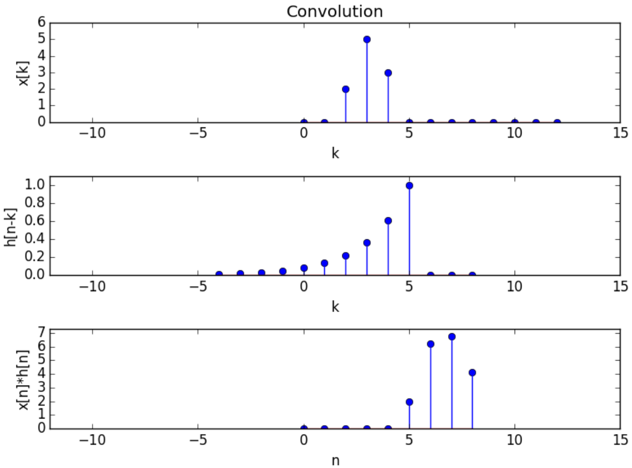




Convolution: Illustration

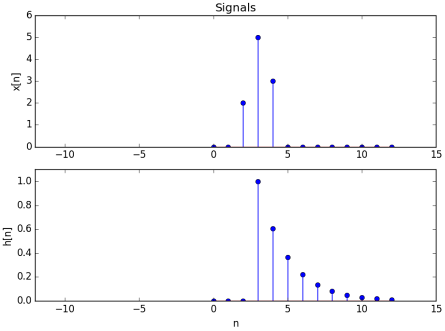
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

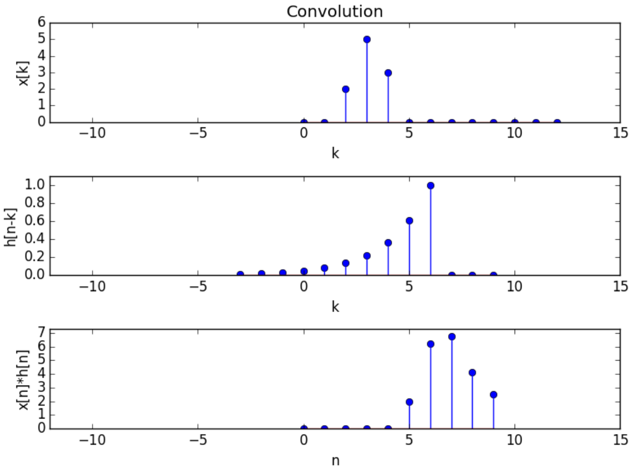




Convolution: Illustration

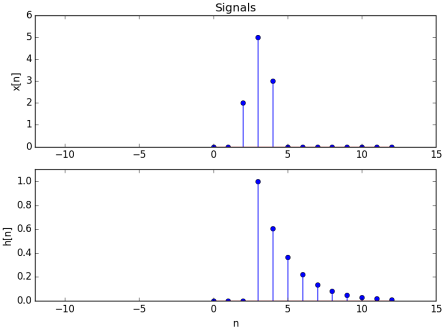
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

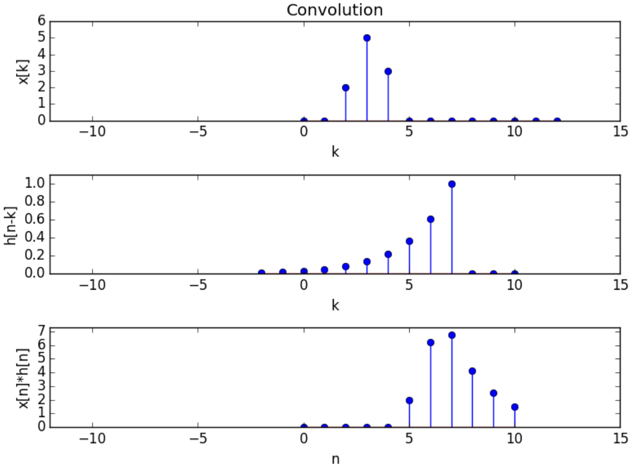




Convolution: Illustration

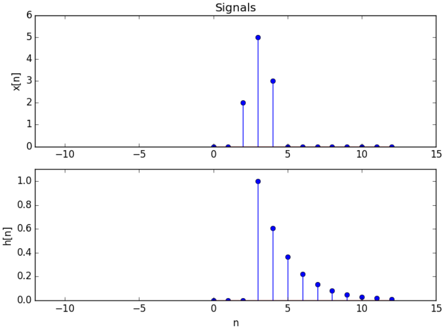
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

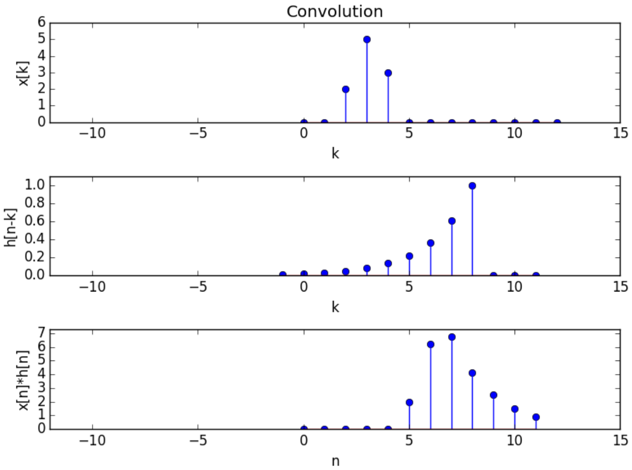




Convolution: Illustration

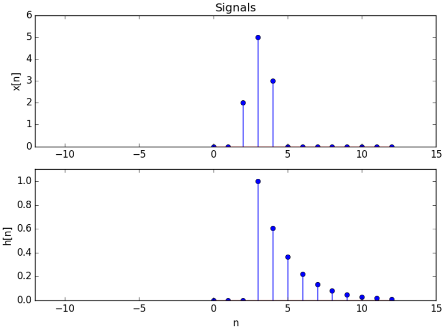
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

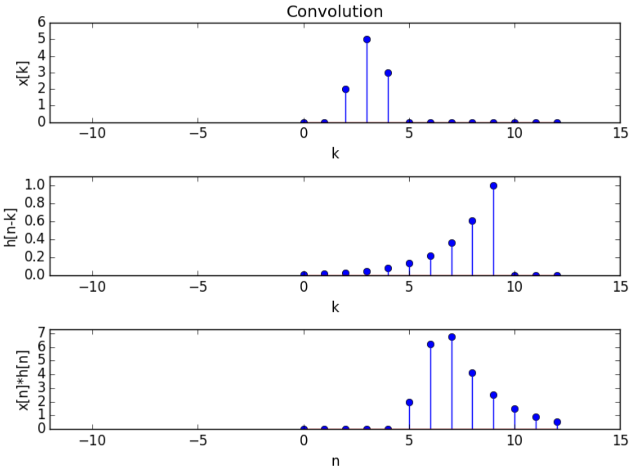




Convolution: Illustration

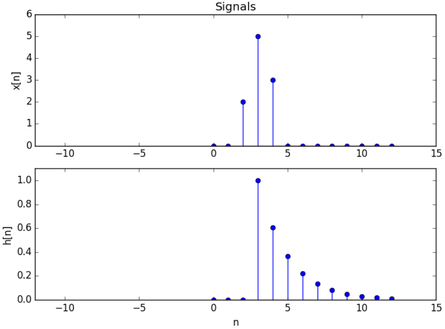
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

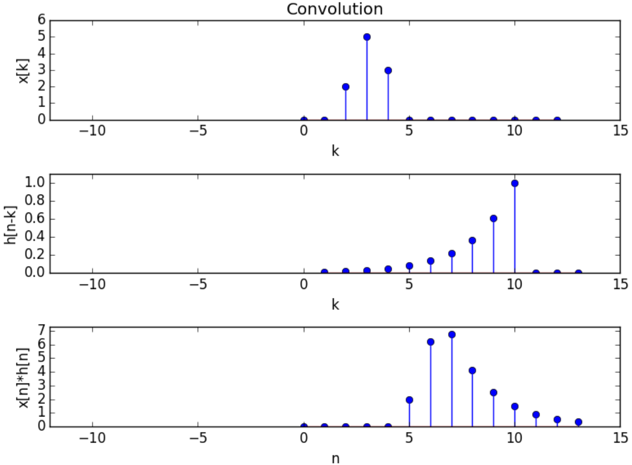




Convolution: Illustration

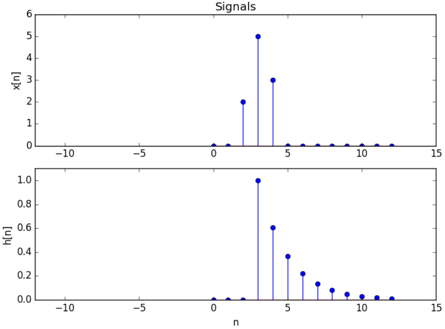
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

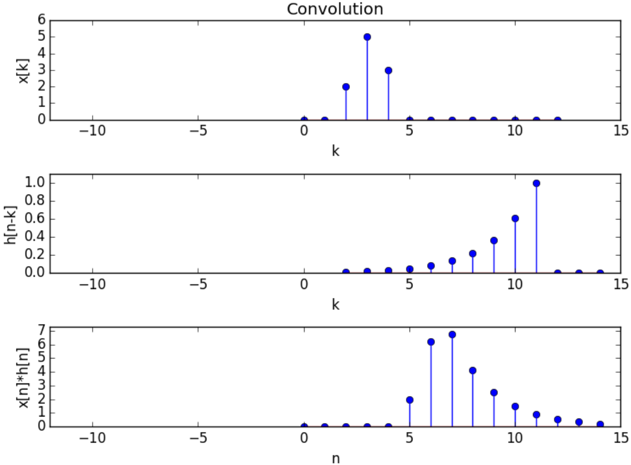




Convolution: Illustration

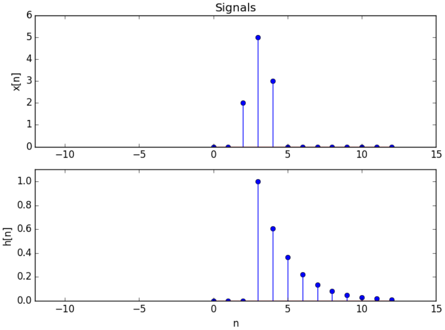
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

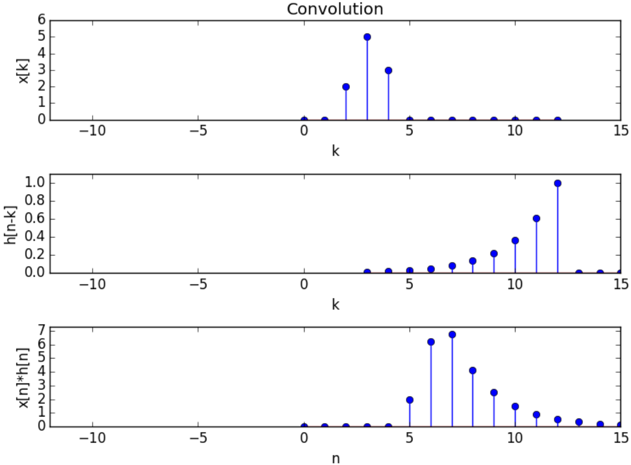




Convolution: Illustration

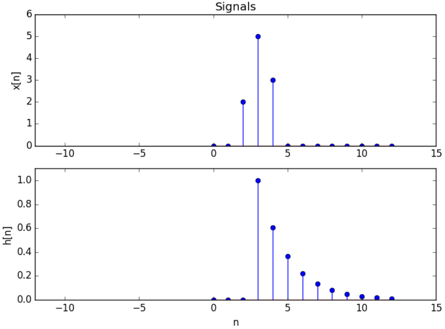
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

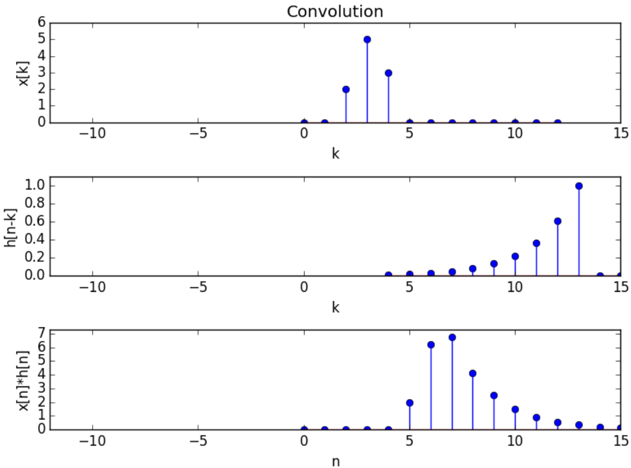




Convolution: Illustration

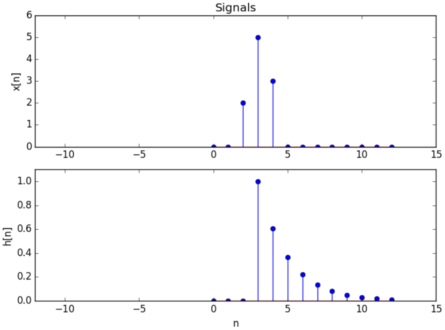
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

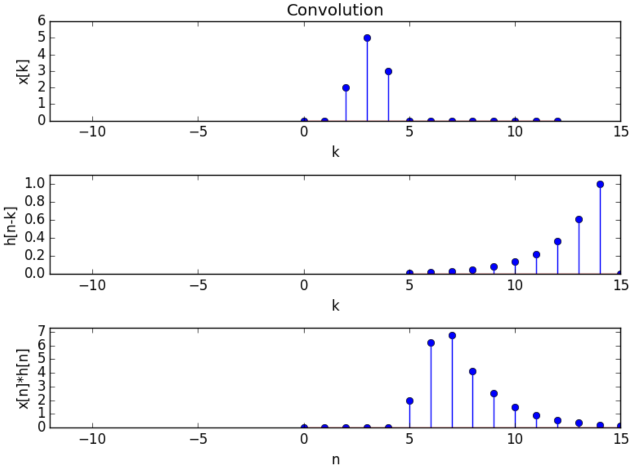




Convolution: Illustration

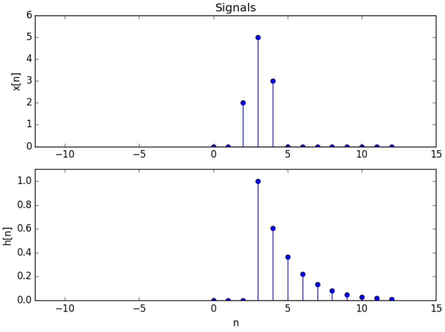
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

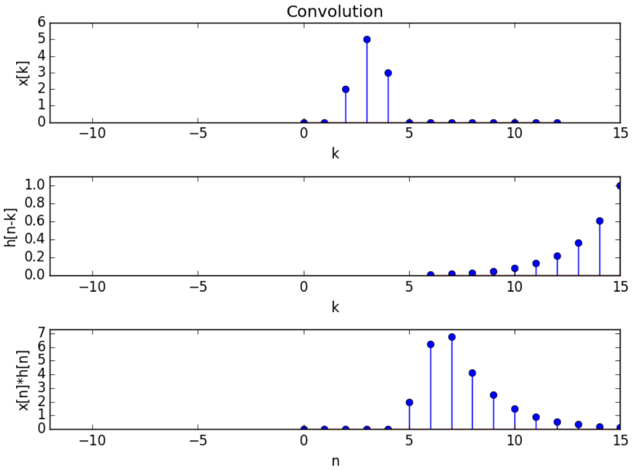




Convolution: Illustration

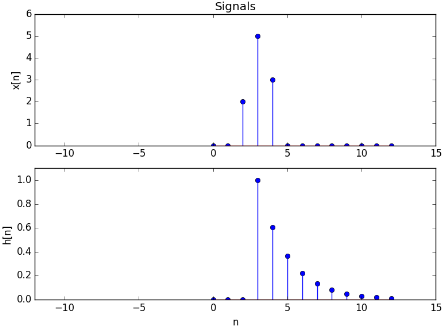
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

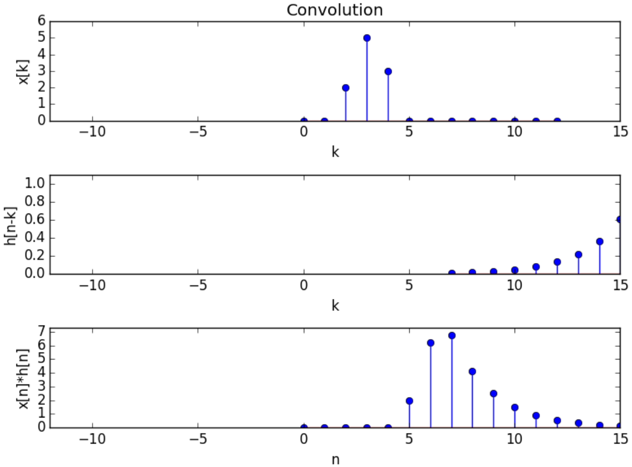




Convolution: Illustration

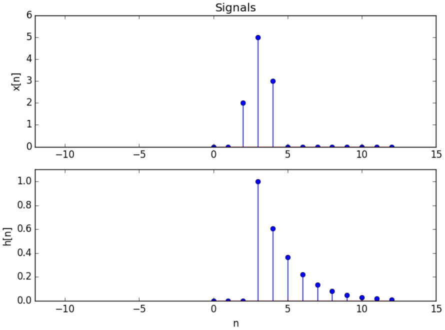
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

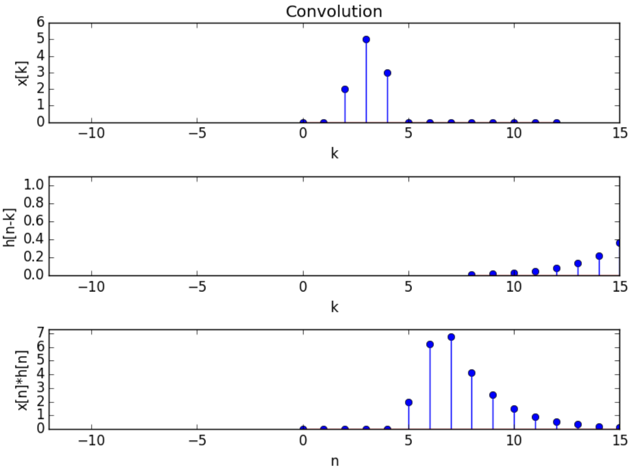




Convolution: Illustration

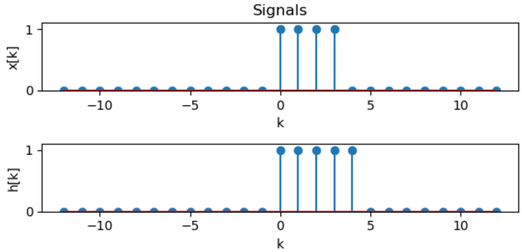
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



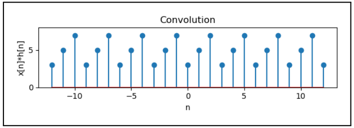


Question: Which is the convolution of the signals

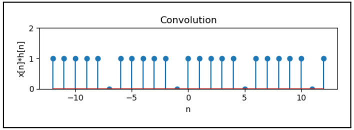
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



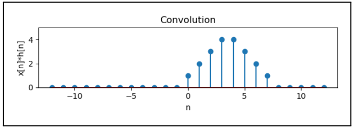
a)



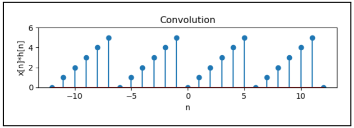
b)



c)

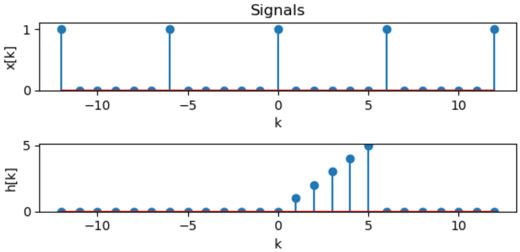


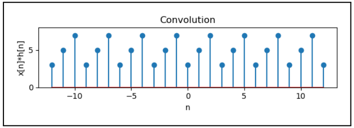
d)

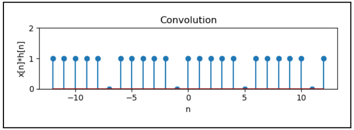


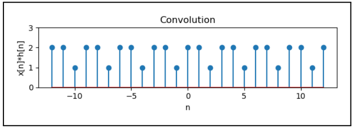
Question: Which is the convolution of the signals

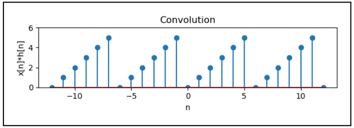
$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



a) 

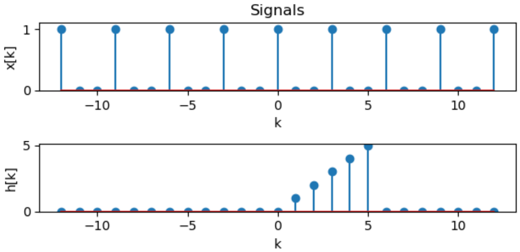
b) 

c) 

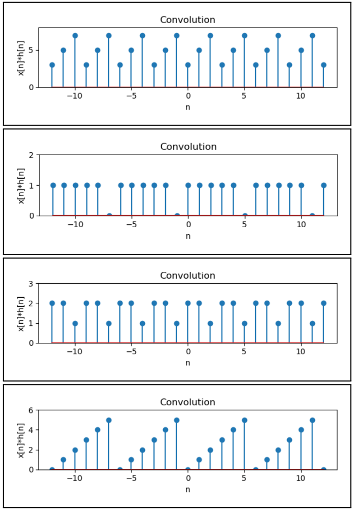
d) 

Question: Which is the convolution of the signals

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$



a)
b)
c)
d)



Linear filter in general (Infinite Impulse Response)

scipy.signal.lfilter(b, a, x, . . .)

$$\begin{aligned} y[n] &= \frac{1}{a_0}\Big(b_0 x[n] + b_1 x[n-1] + \cdots + b_P x[n-P] + \\ &\quad -a_1 y[n-1] - a_2 y[n-2] - \cdots + a_Q y[n-Q]\Big) \\ &= \frac{1}{a_0}\left(\sum_{i=0}^{P} b_i x[n-i] - \sum_{j=1}^{Q} a_j y[n-j]\right) \end{aligned}$$

$$\begin{aligned} a &= [a_0, a_1, \ldots, a_Q] \\ b &= [b_0, b_1, \ldots, b_P] \end{aligned}$$

Fourier Transforms

Fourier transform of continuous signals

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt$$

Fourier transform of discrete signals

$$X(\omega) = \sum_{k=-\infty}^{\infty} x[k] e^{-j\omega k}$$

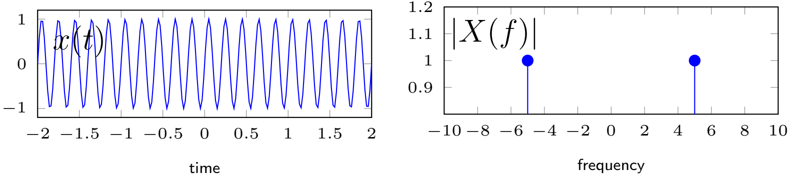
Discrete Fourier Transform (and Fast Fourier Transform)

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

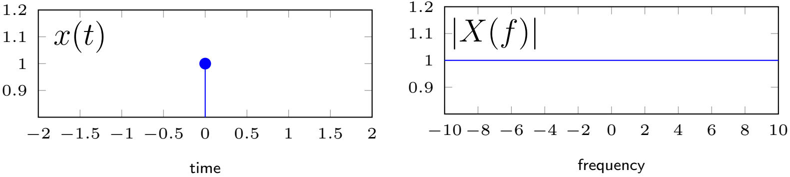
Properties of Fourier Tranform

Property	Time domain		Frequency domain
Linearity	$ax[n] + by[n]$	$\Longleftrightarrow$	$aX(\omega) + bY(\omega)$
Time-shift	$x[n-k]$	$\Longleftrightarrow$	$X(\omega)e^{-j\omega n T_s}$
Convolution	$x[n] * h[n]$	$\Longleftrightarrow$	$X(\omega)H(\omega)$
Multiplication	$x[n]w[n]$	$\Longleftrightarrow$	$X(\omega) * W(\omega)$

Sinusoidal at frequency $f = 5Hz$

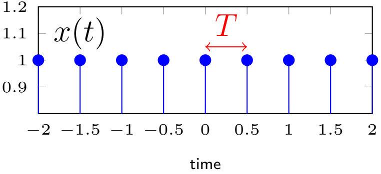


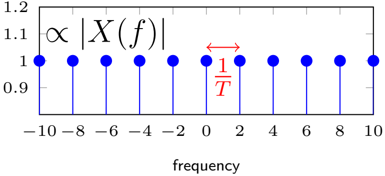
Single (Ideal) Impulse



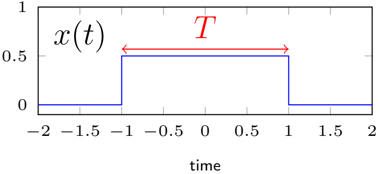
Fourier Transform of Useful Signals

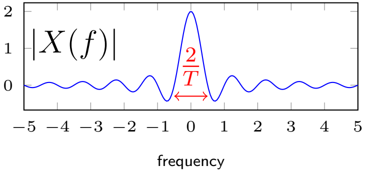
Train of (Ideal) Impulses





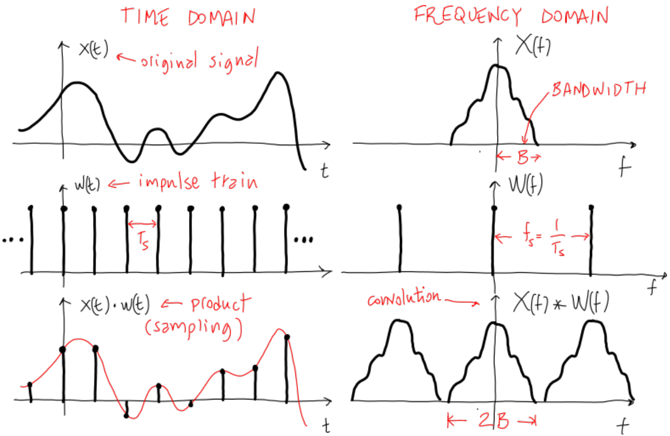
Square function





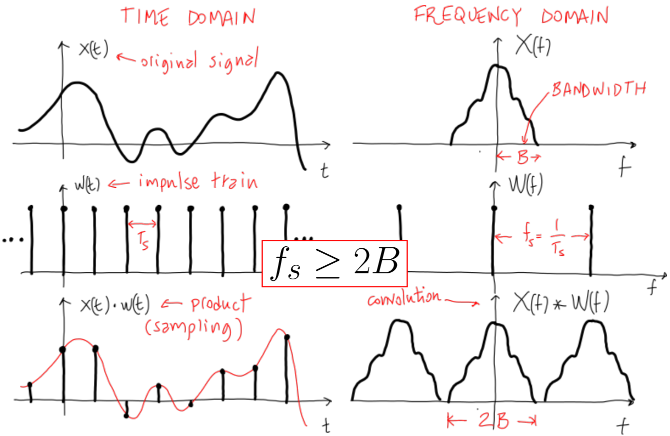
Properties of Fourier Transform — Example: sampling

Time domain		Frequency domain
$x[n]w[n]$	$\Longleftrightarrow$	$X(\omega) * W(\omega)$



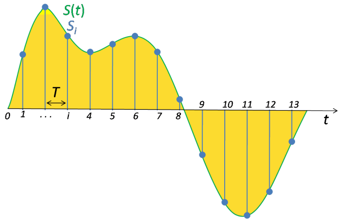
Properties of Fourier Transform — Example: sampling

Time domain		Frequency domain
$x[n]w[n]$	$\Longleftrightarrow$	$X(\omega) * W(\omega)$



$$\boxed{f_s \geq 2B}$$

Sampling Theorem (Nyquist-Shannon)



$$x(t), t \in \mathbb{R} \longrightarrow \boxed{\text{sampling}} \longrightarrow x[n], n \in \mathbb{N}$$

If $x(t)$ contains energy up to B_x, in order to reconstruct the signal we need to sample with

$$f_s > 2B_x$$

Aliasing

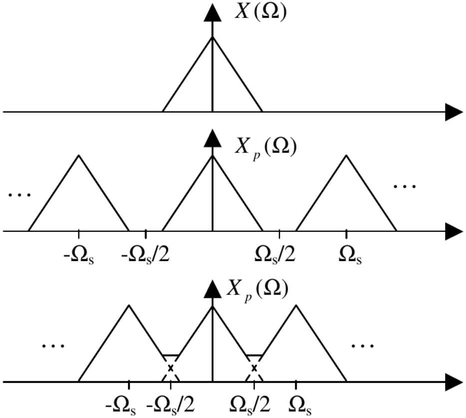


Figure from Huang, Acero and Hon (2001)

Aliasing: Illustration

Video from `https://youtu.be/usN47Jvy9PY` (unfortunately removed)

First step: represent speech signal

Sampling

- Nyquist-Shannon Theorem: sample at twice the band
- 8kHz (4kHz band, telephone), 16kHz (8 kHz band, high quality)
- TIDIGITS sampled at 20kHz
- TIMIT sampled at 16kHz

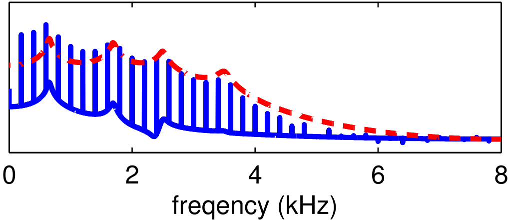
Quantisation

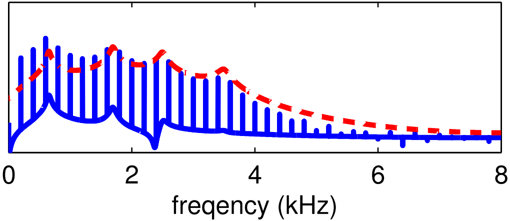
- Type of quantisation: linear, a-law, μ-law
- 8, 16 bits (more rare 32, floating point)
- TIDIGITS and TIMIT are quantised with 16 bits linear

Pre-emphasis

Compensate for the 6db/octave drop (glottal shape - radiation at the lips)

$$y[n] = x[n] - \alpha x[n-1]$$



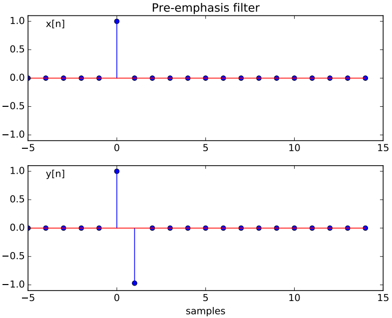


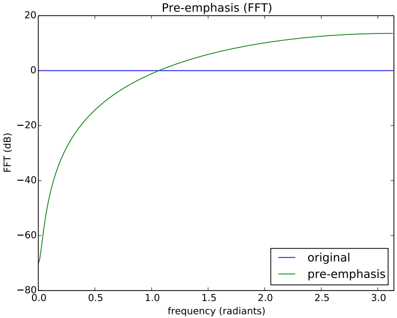
α is usually 0.95–0.97

Pre-Emphasis as Linear Filter

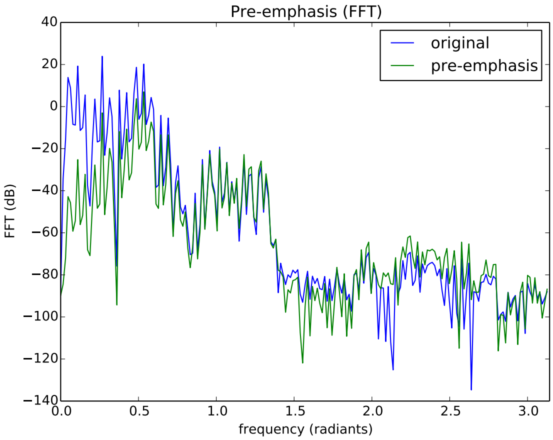
Time domain		Frequency domain
$x[n] * h[n]$	$\Longleftrightarrow$	$X(\omega)H(\omega)$

$$y[n] = x[n] - \alpha x[n-1], \quad \text{with } \alpha = 0.97$$

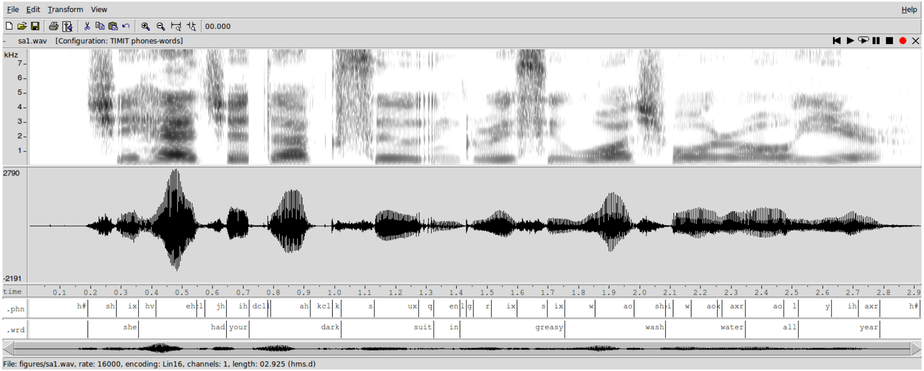




Pre-emphasis applied to vowel

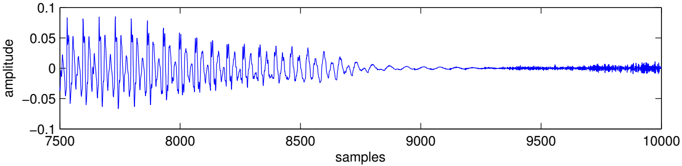


A time varying signal

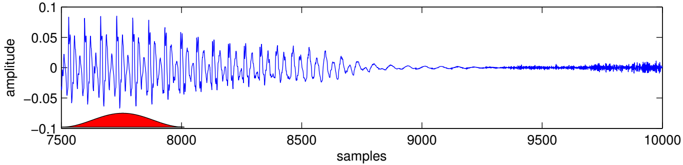


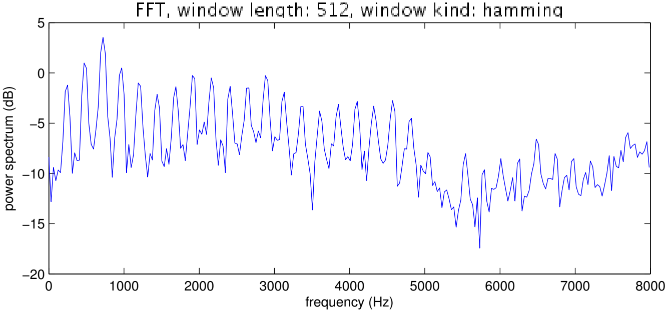
- speech is time varying
- short segment are quasi-stationary
- use short time analysis

Short-Time Fourier Analysis

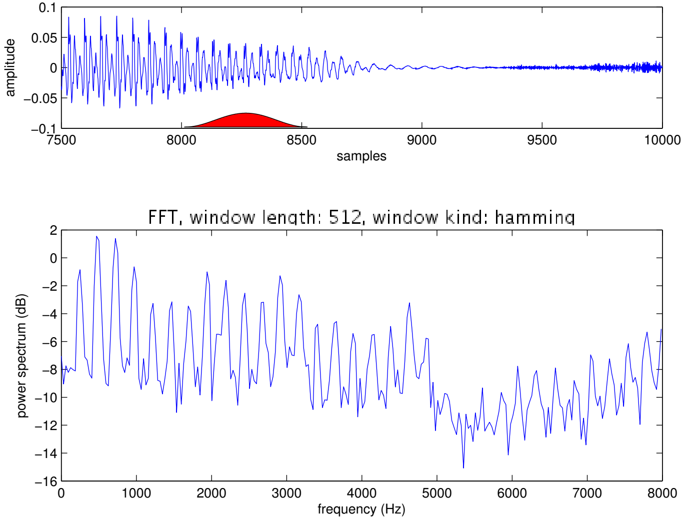


Short-Time Fourier Analysis

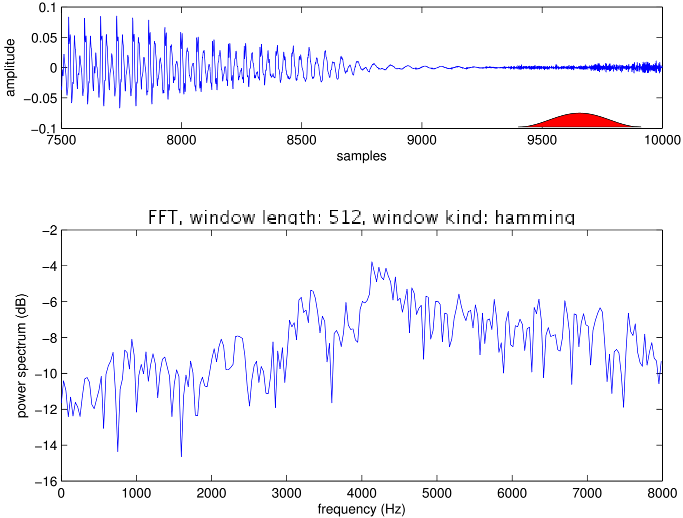




Short-Time Fourier Analysis

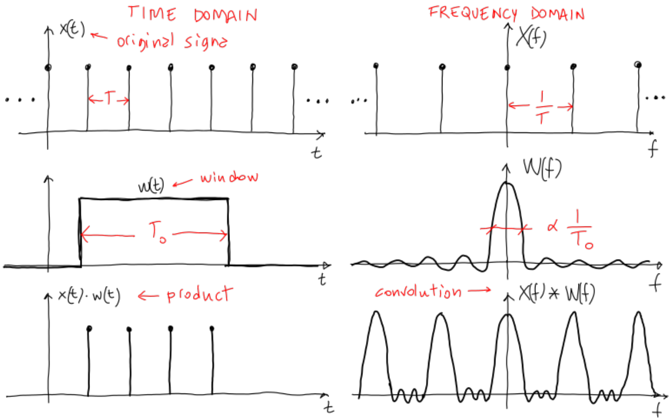


Short-Time Fourier Analysis



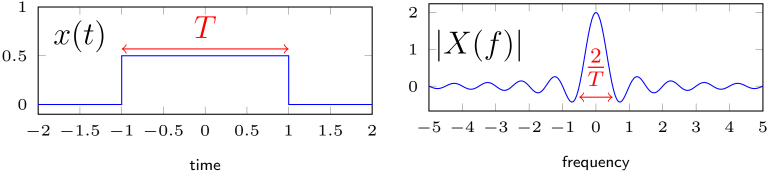
Properties of Fourier Transform — Example: Windowing

Time domain		Frequency domain
$x[n]w[n]$	$\Longleftrightarrow$	$X(\omega) * W(\omega)$

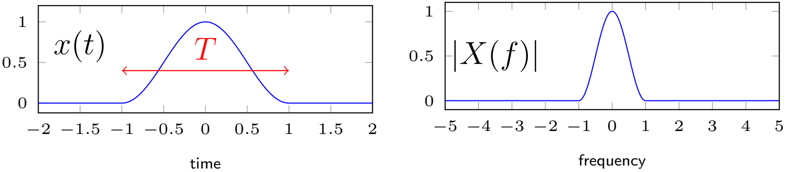


Effect of Windowing

Square window

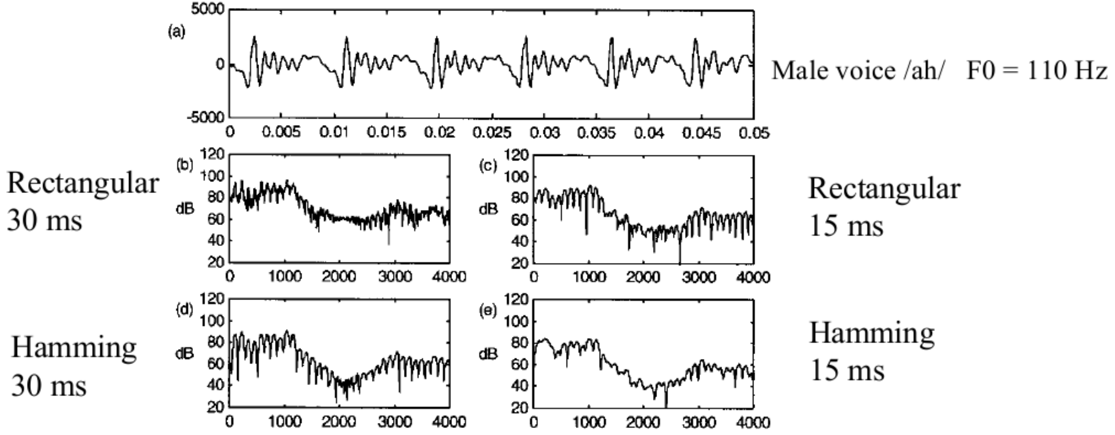


Hamming window



Effect of Windowing on Speech

Effect of different window functions



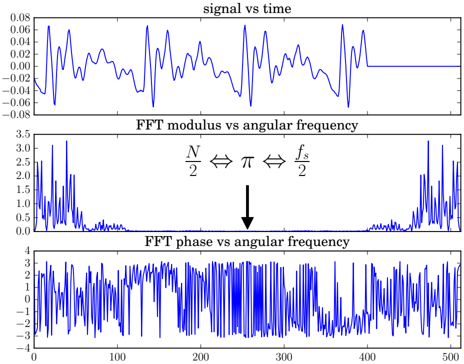
Windowing, typical values

- signal sampling frequency: 8–20kHz
- analysis window: 10–50ms
- frame step: 10–25ms (100–40Hz)

Fast Fourier Transform (FFT) ($N = 512$)

scipy.fftpack.fft(x, n=512, ...)

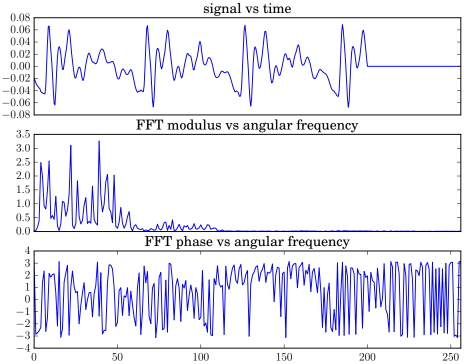
$$X[n] = \sum_{k=0}^{N-1} x[k] e^{-j2\pi \frac{n}{N} k}$$



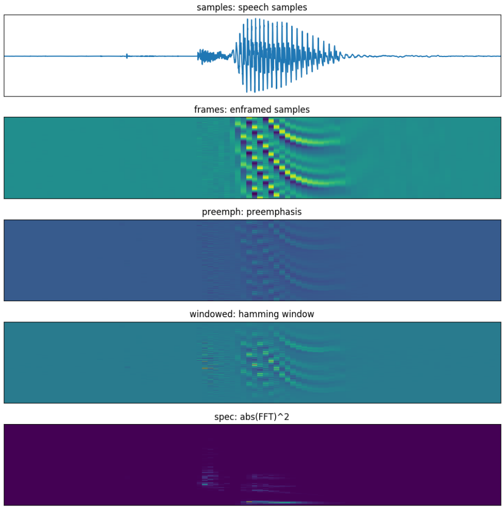
Fast Fourier Transform (FFT) ($N = 512$)

scipy.fftpack.fft(x, n=512, ...)

$$X[n] = \sum_{k=0}^{N-1} x[k] e^{-j2\pi \frac{n}{N} k}$$



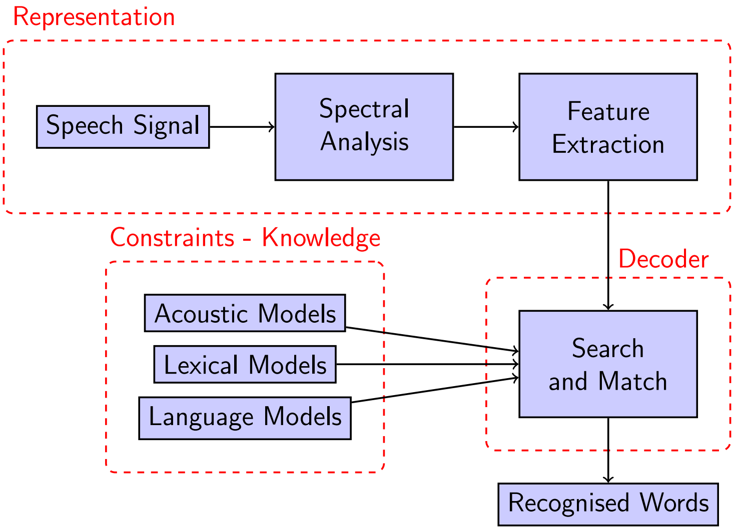
Speech signal processing in practice



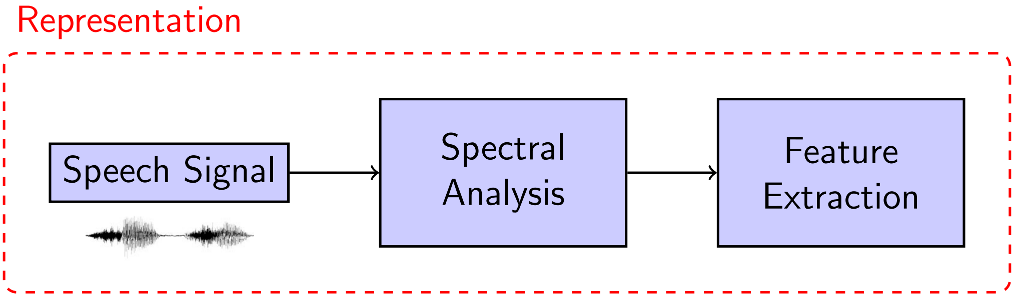
Outline

1. Speech Signal Representations
   - Signal Processing Reminder
   - Sampling and Quantization
   - Pre-Emphasis
   - Windowing
   - Discrete Fourier Transform
2. Feature Extraction
   - Linear Prediction Analysis (LPA)
   - Cepstrum
   - Perceptually Motivated Features

Components of ASR System

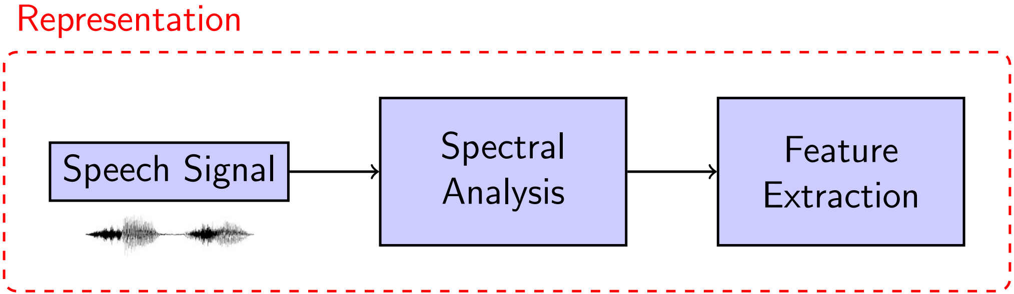


Speech Signal Representations



Goals:

- disregard irrelevant information
- optimise relevant information for modelling

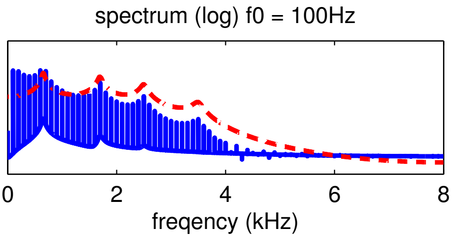


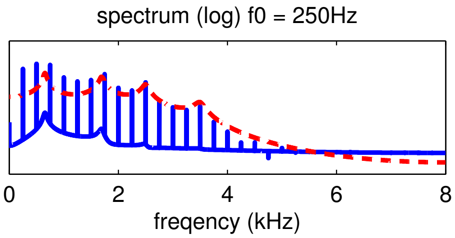
Means:

- try to model essential aspects of speech production
- imitate auditory processes
- consider properties of statistical modelling

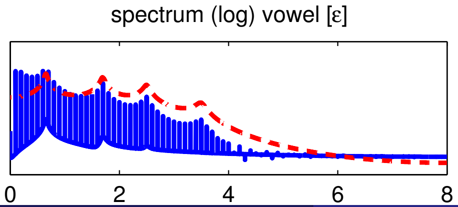
F_0 and Formants

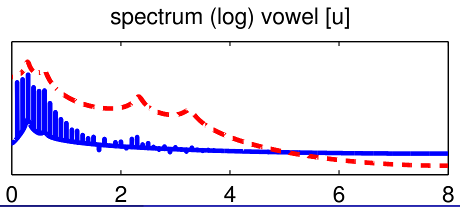
- Varying F_0 (vocal fold oscillation rate)



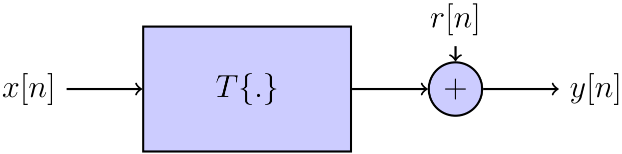


- Varying Formants (vocal tract shape)





Linear Prediction Coefficients (LPC)



approximate $y[n]$ as a linear combination of p previous samples:

$$\hat{y}[n] = \sum_{k=1}^{p} a_k y[n-k]$$

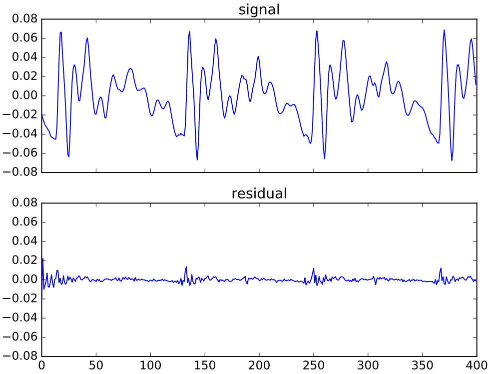
The error is called residual: $r[n] = \hat{y}[n] - y[n]$
The output of the signal is:

$$y[n] = \sum_{k=1}^{p} a_k y[n-k] + r[n]$$

LPC and Speech coding

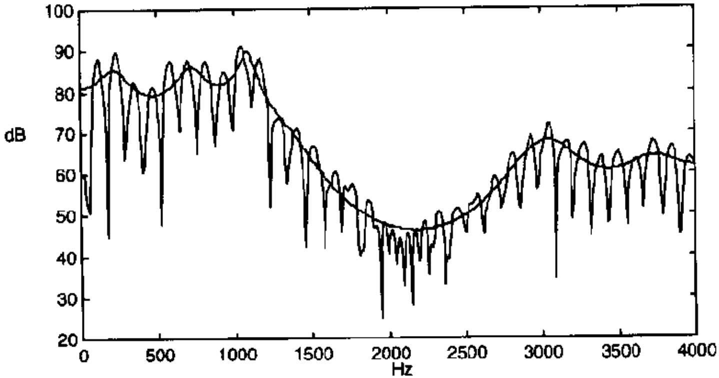
$$y[n] = \sum_{k=1}^{p} a_k y[n-k] + r[n]$$

- We only need to send $a_1, \ldots, a_p$ and $r[n]$.
- $r[n]$ can be coded with fewer bits



LPC Example

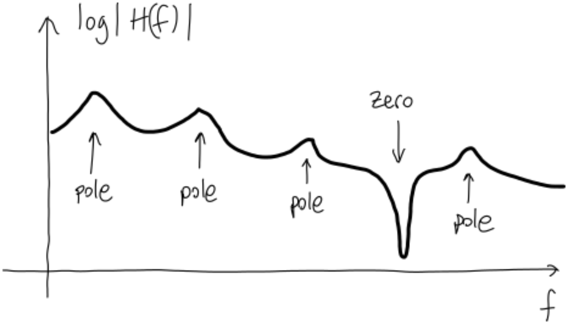
$$y[n] = \sum_{k=1}^{p} a_k y[n-k] + r[n]$$



Infinite Impulse Response (IIR) Systems

In general y depends on (delayed) samples of the input, as well as the output at previous times (feedback)

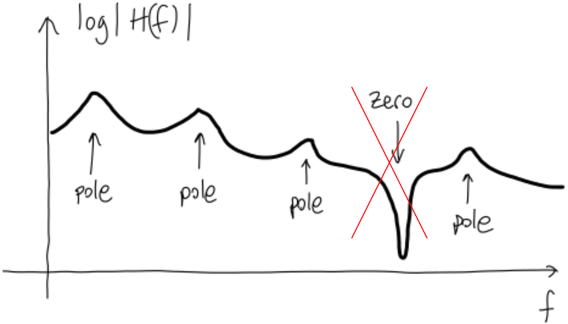
$$
\begin{aligned}
y[n] \quad = \quad & \frac{1}{a_0} \sum_{h=0}^{P} b_h x[n-h] & \leftarrow \text{zeros} \\
& - \frac{1}{a_0} \sum_{k=1}^{Q} a_k y[n-k] & \leftarrow \text{poles}
\end{aligned}
$$



Linear Prediction

Auto regressive (AR): only depends on current input and the output at previous times (feedback)

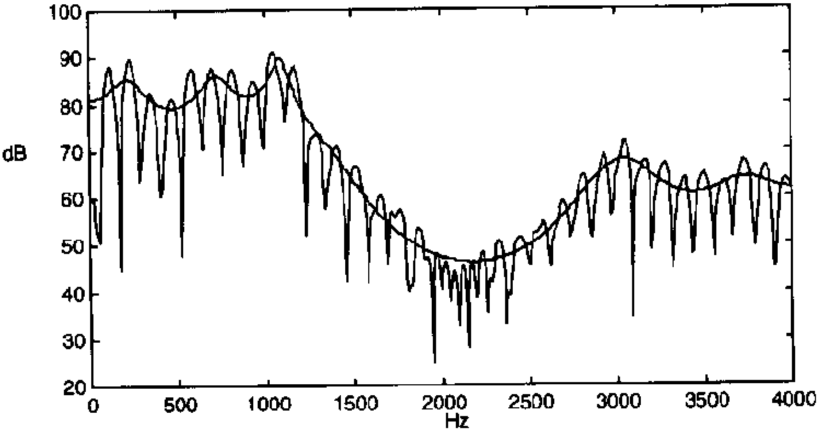
$$
\begin{aligned}
y[n] &= \frac{b_0}{a_0} x[n] && \leftarrow \text{no zeros} \\
&\quad - \frac{1}{a_0} \sum_{k=1}^{Q} a_k y[n-k] && \leftarrow \text{poles}
\end{aligned}
$$



See also poles_and_zeros.pdf in Blackboard

LPC Limitations

- better match at spectral peaks than at valleys (all-pole model)
- not accurate if transfer function contains zeros (nasals, fricatives. . .)



Cepstrum Rationale (Homomorphic Transformation)

- signals combined in a convolutive way: $a[n] * b[n] * c[n]$
- in the spectral domain: $A(z)B(z)C(z)$
- taking the $\log$: $\log(A(z)) + \log(B(z)) + \log(C(z))$
- to analise the different contribution perform Fourier transform (DCT if not interested in phase information).

Cepstrum Rationale (Homomorphic Transformation)

- signals combined in a convolutive way: $a[n] * b[n] * c[n]$
- in the spectral domain: $A(z)B(z)C(z)$
- taking the $\log$: $\log(A(z)) + \log(B(z)) + \log(C(z))$
- to analise the different contribution perform Fourier transform (DCT if not interested in phase information).
- Terminology:
  - frequency vs quefrency
  - spectrum vs cepstrum
  - filter vs lifter
  - ...

Cepstrum Definition

Complex Cepstrum

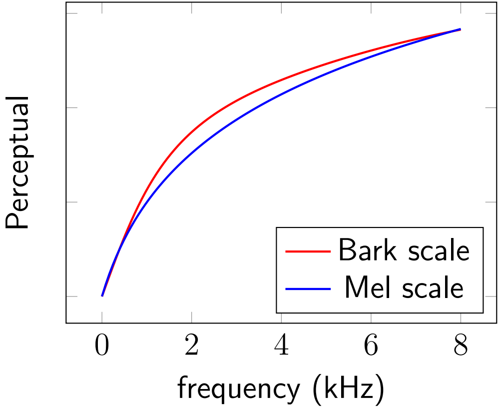
$$\hat{x}[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln X(\omega) e^{j\omega n} d\omega$$

Real Cepstrum

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

Mel Filterbank: Motivation

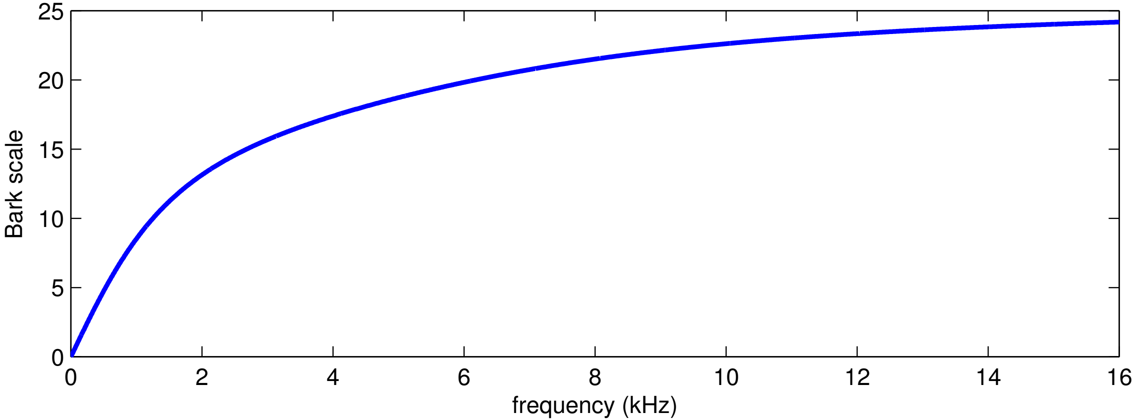
- Perception of frequencies is logarithmic: Mel and Bark scales
- Perception of amplitude (or energy, or loudness) is logarithmic



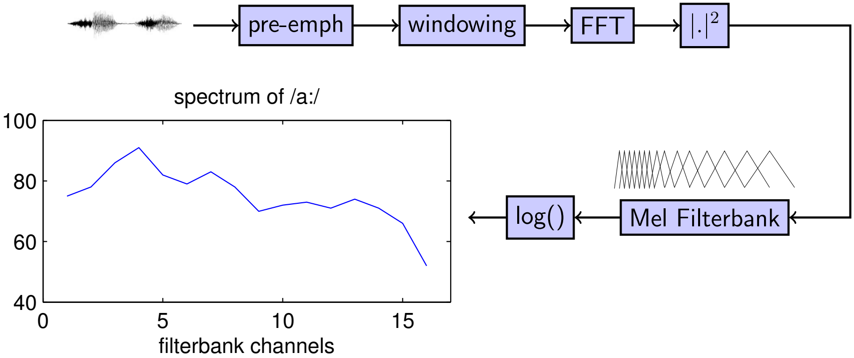
$$\begin{aligned} \text{bark}(f) &= 13\arctan(0.00076 * f) + \\ &\quad +3.5\arctan\left[\left(\frac{f}{7500}\right)^2\right] \\ \text{mel}(f) &= 1125\ln\left(1+\frac{f}{700}\right) \end{aligned}$$

Perceptual Linear Prediction

- Transform to the Bark frequency scale before computing the LPC coefficients
- Cubic root of energy instead of logarithm



Mel Filterbank: Calculation



Mel Filterbank: Pros and Cons

Cons:

- coefficients are correlated (inefficient use of information)
- difficult to model statistically (e.g. multivariate Gaussian distribution)

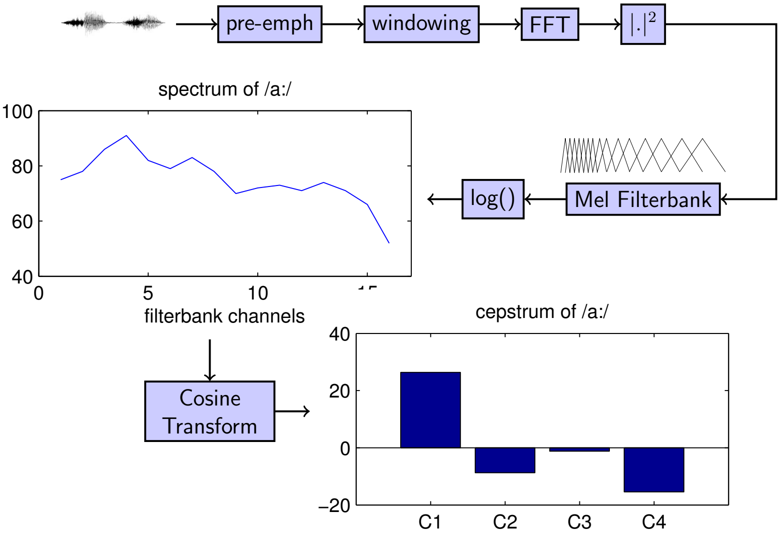
Pros:

- easy to interpret (smooth spectrum)
- works well with neural networks (no problem with correlations)

Mel Frequency Cepstrum Coefficients

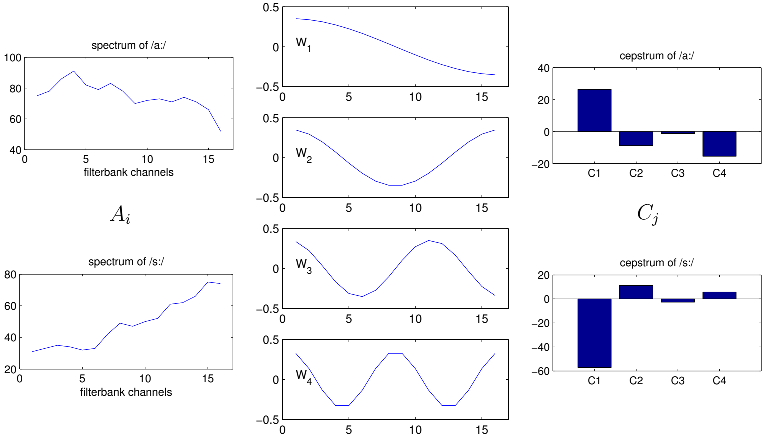
- *de facto* standard in ASR
- imitate aspects of auditory processing
- does not assume all-pole model of the spectrum
- uncorrelated: easier to model statistically

MFCCs Calculation



MFCC: Cosine Transform (scipy.fftpack.realtransforms.dct)

$$C_j = \sqrt{\frac{2}{N}} \sum_{i=1}^{N} A_i \cos\left(\frac{j\pi(i-0.5)}{N}\right)$$



MFCC Advantages

- fairly uncorrelated coefficients (simpler statistical models)
- high phonetic discrimination (empirically shown)
- do not assume all-pole model
- low number of coeff. enough to capture coarse structure of spectrum
- Cepstral Mean Subtraction corresponds to channel removal

Dynamic Features

Concatenate static MFCCs (or LPCs) to Δ and $\Delta\Delta$ vectors.
Δ_n computed as weighted sum of $d_k(n)$

$$\Delta_n = \frac{\sum_{k=1}^{K} w_k d_k(n)}{\sum_{k=1}^{K} w_k}$$

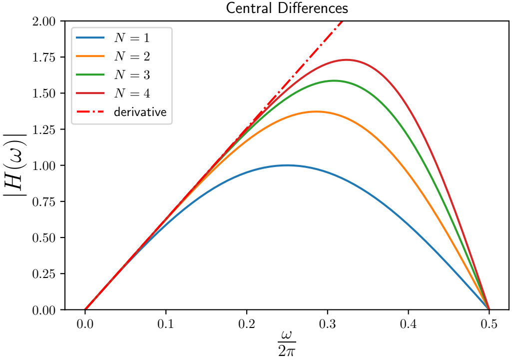
$d_k(n)$: finite differences centered around n with interval $2k$:

$$\begin{aligned} d_k(n) &= \frac{c_{n+k} - c_{n-k}}{2k} \\ w_k &= 2k^2 \end{aligned}$$

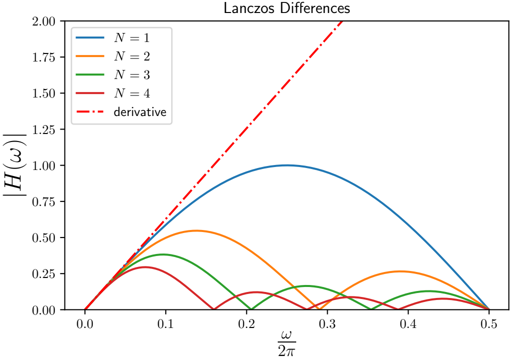
Similarly for $\Delta\Delta_n$

Dynamic Features: Motivation

Central Differences



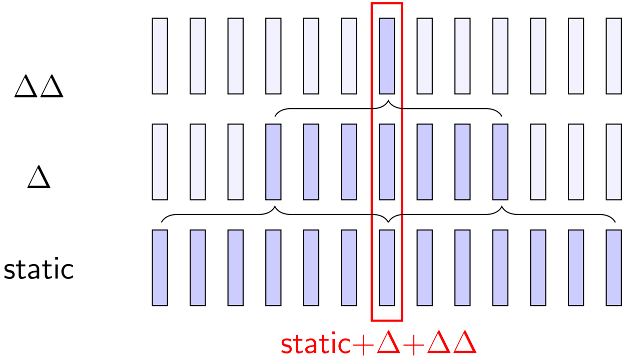
Lanczos Differences



Polynomial fit with or without error

Dynamic Features: Common values

- Usually k goes from 1 to 3
- to compute static+Δ+$\Delta\Delta$ we need 13 consecutive static vectors (around 130 msec).



MFCCs: typical values

- 12 Coefficients C1–C12
- Energy (could be C0)
- Delta coefficients (derivatives in time)
- Delta-delta (second order derivatives)
- total: 39 coefficients per frame (analysis window)