Tampere University

Time-Frequency Representations

SGN 14007

Lecture 3

Annamaria Mesaros

Spectral analysis

- Representing signals in spectral domain
- Decomposition into their frequency components
  - Similar to human ear analysis
- Fourier analysis! (Fourier transform)
  - Used in a large variety of signal processing techniques (not just audio!)
  - Decomposition of the signal into basic functions
- Spectrum analysis = (typically) magnitude spectrum
  - Because auditory system is relatively insensitive to phase!
- dB scale to mimic perception of the signal level (mostly)

Time-frequency representations

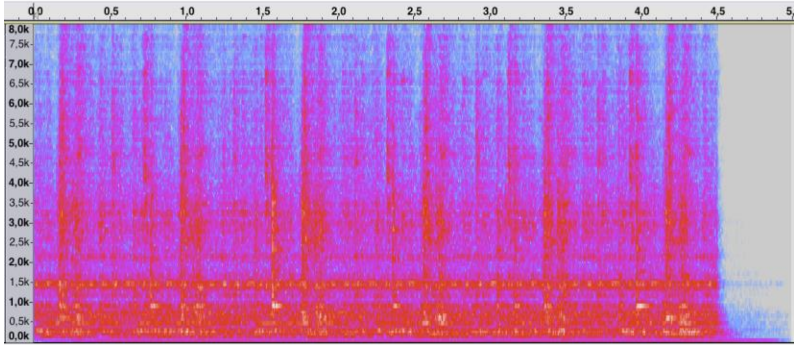
- Represent signal changes in **both time and frequency** at the same time!
- Spectrogram
- Time-frequency analysis
  - Allows separate processing of the signal in different frequency bands according to time
  - Allows processing modeled based on human hearing mechanisms

Short-time transforms: use frame-based processing

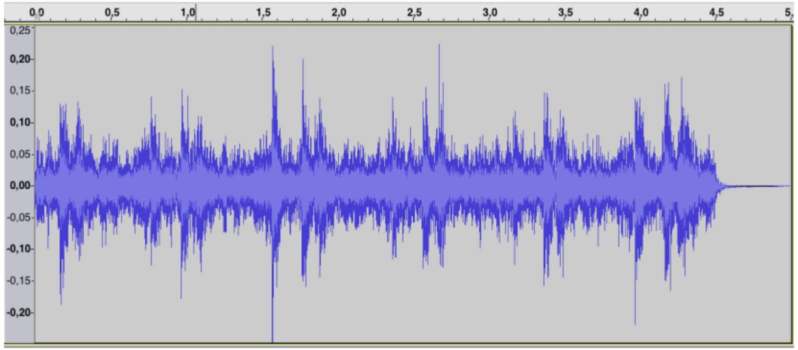
- Sequential processing of small time frames
  - Assumes signal is time-invariant within this frame length
- Signal is divided into frames: kth frame if input signal is $x(n+Lk)$
  - $n=0,\ldots,N$, where N is window length,
  - $k=0, 1, 2,\ldots$ is window index
  - L is hop size overlapping frames!
- Typical frame length is 20 ms (speech) to 100 ms (music)
- Frame-based processing allows real-time processing for communication applications

Time vs time-frequency representation

Printer noise



Time-frequency representation: Notice noise is stationary in some spectral bands over time.



Time domain: More difficult to verify signal assumptions about noise stationarity

Transforms

- Calculate the inner product between the signal and basis functions of the transform
- Audio signal processing typically uses basis functions that are sines and cosines with different frequencies
- Output: The spectrum of the signal
- Efficient algorithms exist: fast Fourier transform

Naive framing approach: just use signal values within the time frame

- Implicit rectangular window
- Multiplication in time domain = convolution in frequency domain
    - Magnitude response of a rectangular window ($w(n)=1$) is a sinc() function.
    - This can smear the spectrogram and result in reduced resolution

Special window functions $w(n)$ are used: Hamming, Hanning

Discrete Fourier Transform (DFT)

DFT of time domain signal x(n) is (n is sample index)

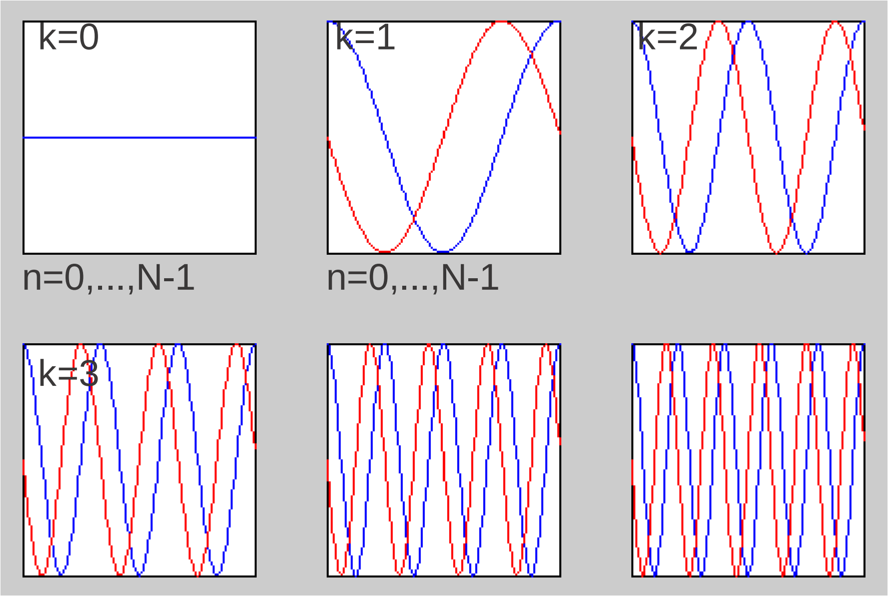
$$X(k) = \sum_{n=0}^{N-1} w_a(n)x(n)\exp(-j2\pi\, kn/N)$$

- *k* is the frequency bin index (max *K*)
- *N* is the window length
- $w_a(n)$ is an analysis window
- $\exp(-j2\pi kn/N)$ is the DFT basis function for frequency k, n=0…N-1

Performing DFT for sequential windows of data is called STFT: **Short-time Fourier transform** k=0

Discrete Fourier transform

First few basis functions(blue:real part, red:imaginary part)

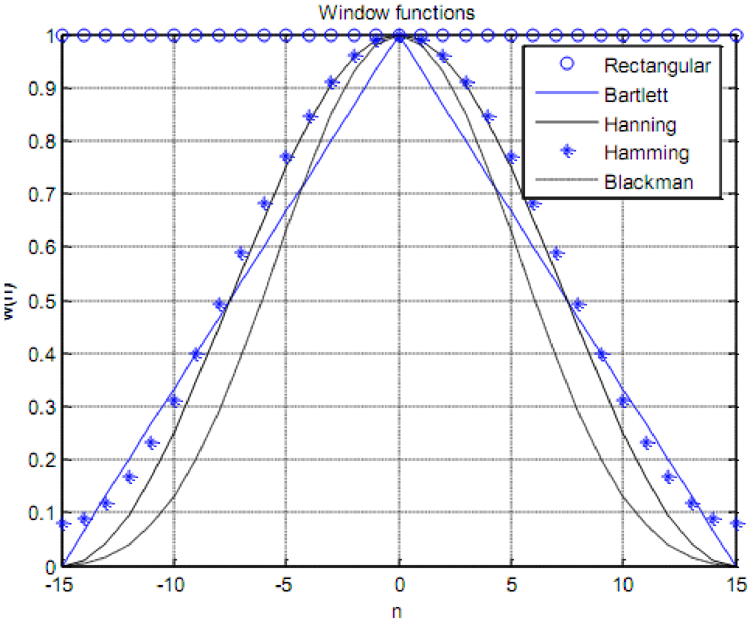


Windowing

For each audio frame, a window is applied:

$$x(n) = x(n) \cdot w(n), n = 0, \ldots, N - 1$$

- *w*(*n*) is the windowing function
- rectangular window *w*(*n*)=[1,1,…,1]



Rectangular vs Hamming window

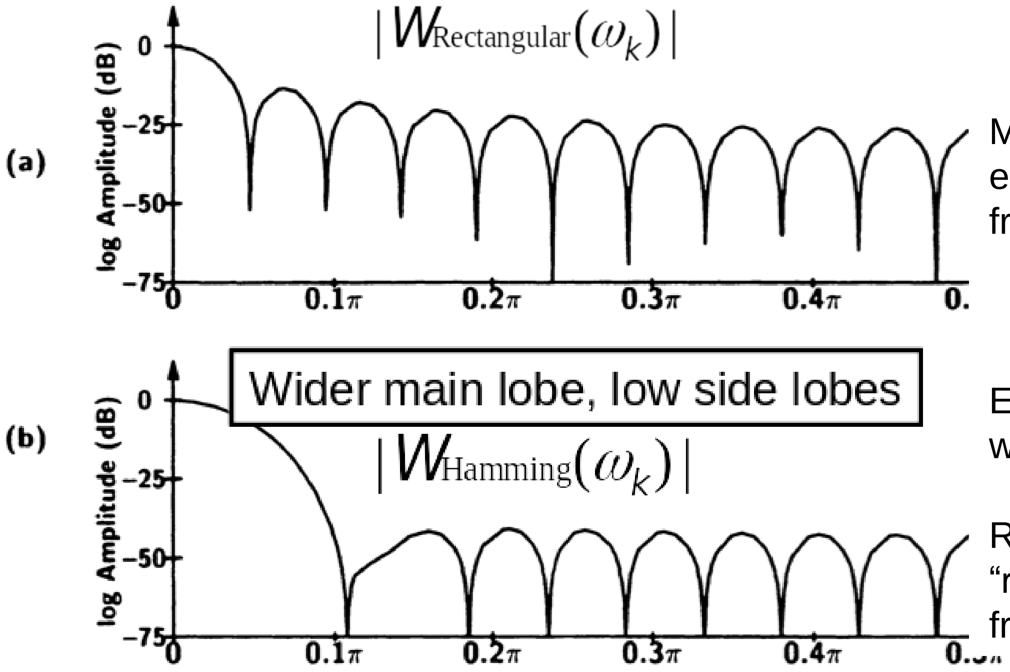


Figure 6.3 Magnitude of Fourier transforms for (a) rectangular window, (b) Hamming window.

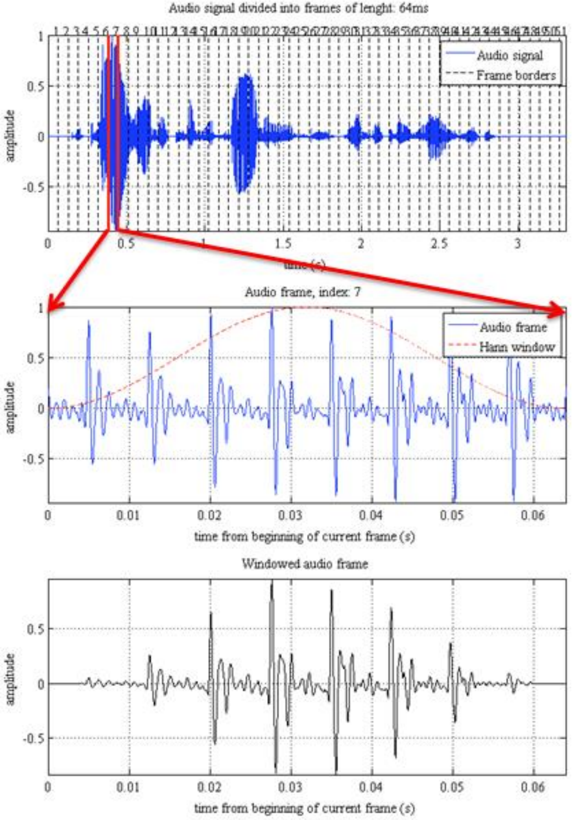
Multiplication in time-domain is equivalent to convolution in frequency domain

$$\tilde{X}(\omega_k) = X(\omega_k) * W(\omega_k)$$

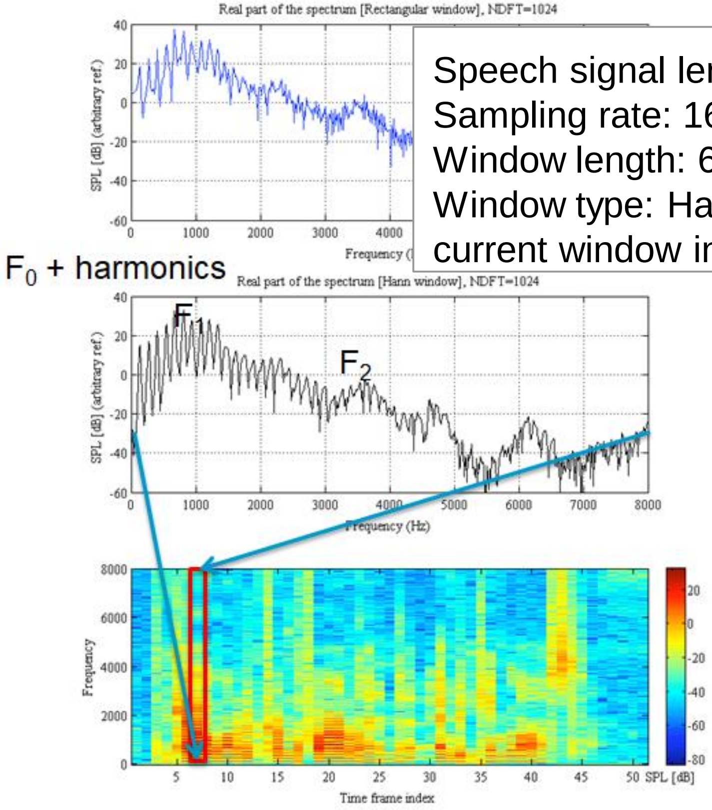
Each spectral line is convolved with the window response.

Rectangular window results in “ringing” (it creates additional frequencies)

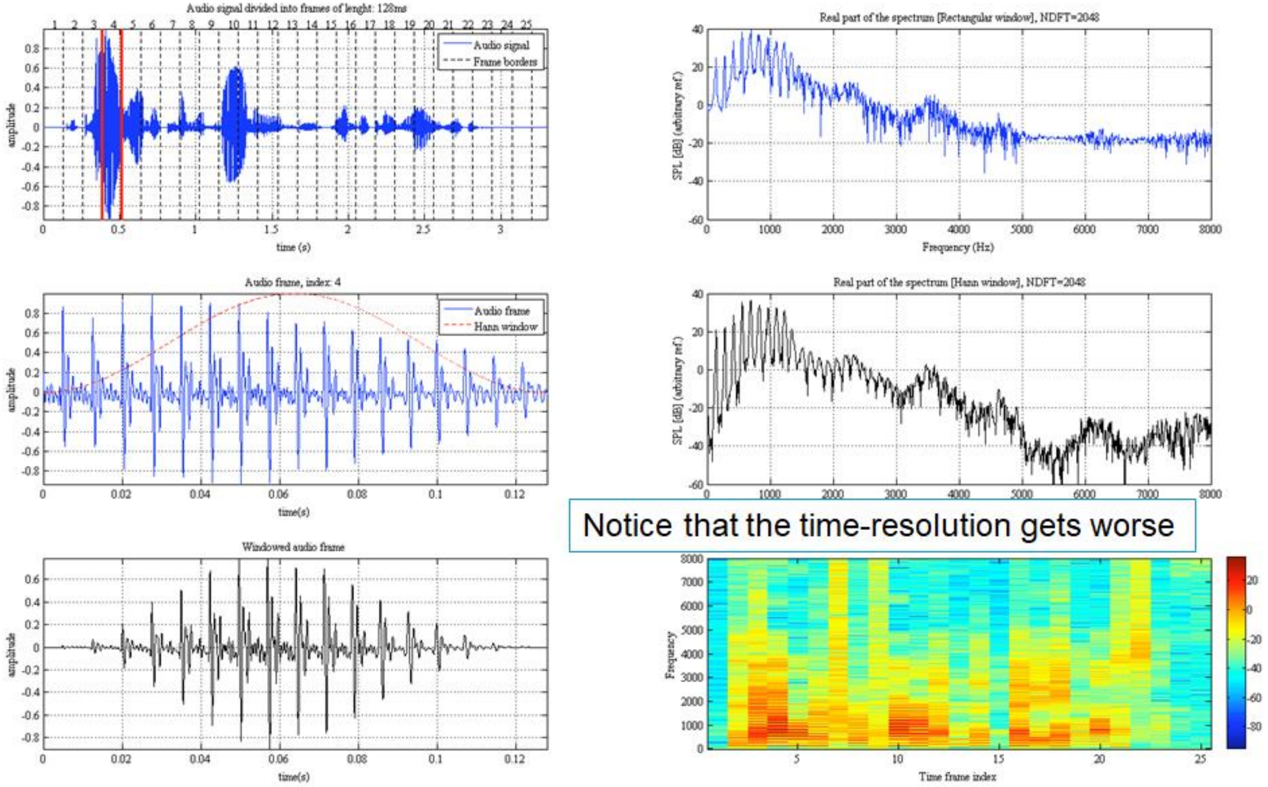
64 ms analysis window



Speech signal length, 3.4s
Sampling rate: 16 kHz
Window length: 64ms
Window type: Hann
current window index: 7

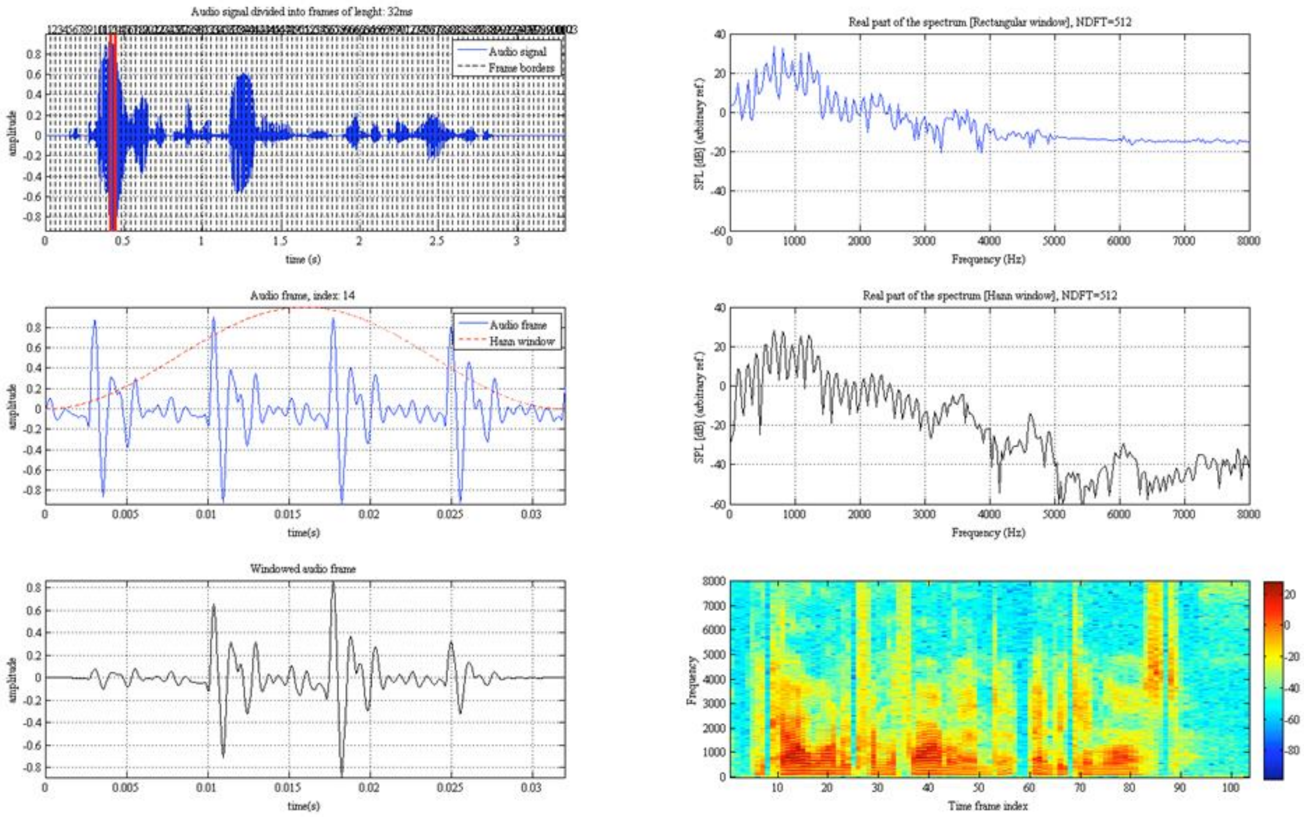


128 ms analysis window

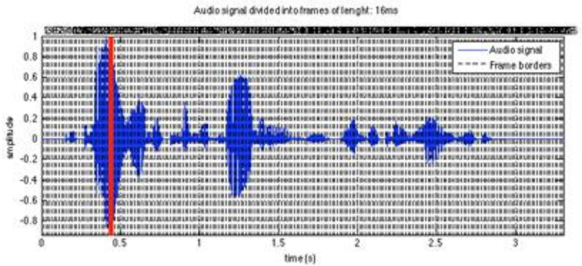


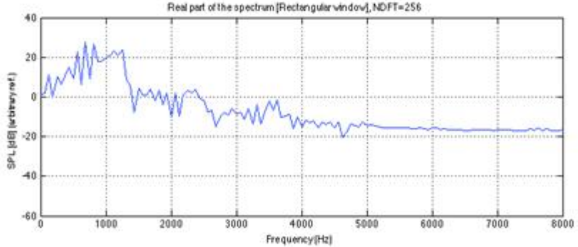
Notice that the time-resolution gets worse

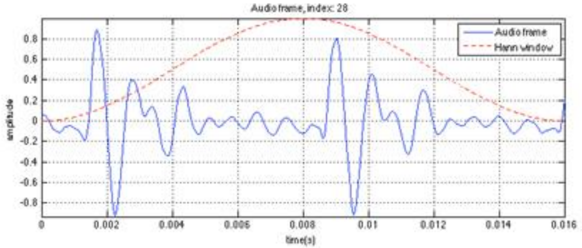
32 ms analysis window

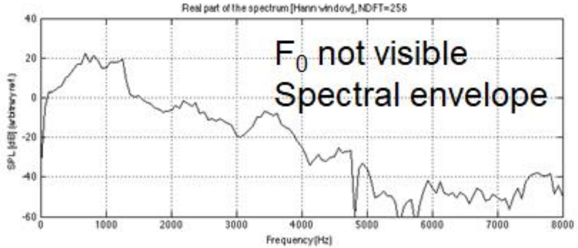


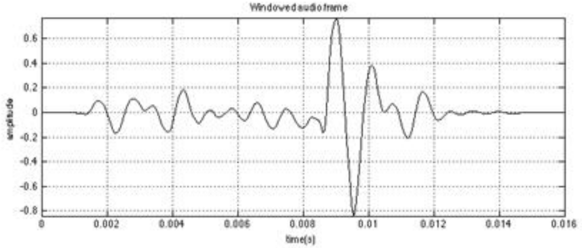
16 ms analysis window

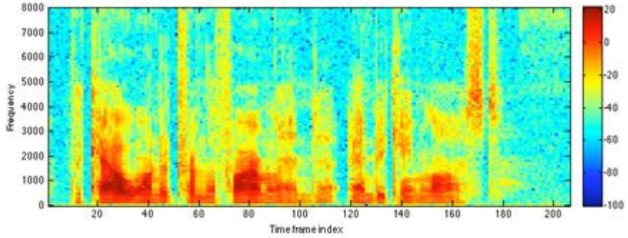












Observations

Increasing the window length

- Spectrogram has better frequency resolution
- But worse time resolution in the STFT

Rectangular windowing

- Spectral smearing

Specific windowing (e.g. Hann)

- Better visibility of spectral peaks.

Thinking break (2 minutes)

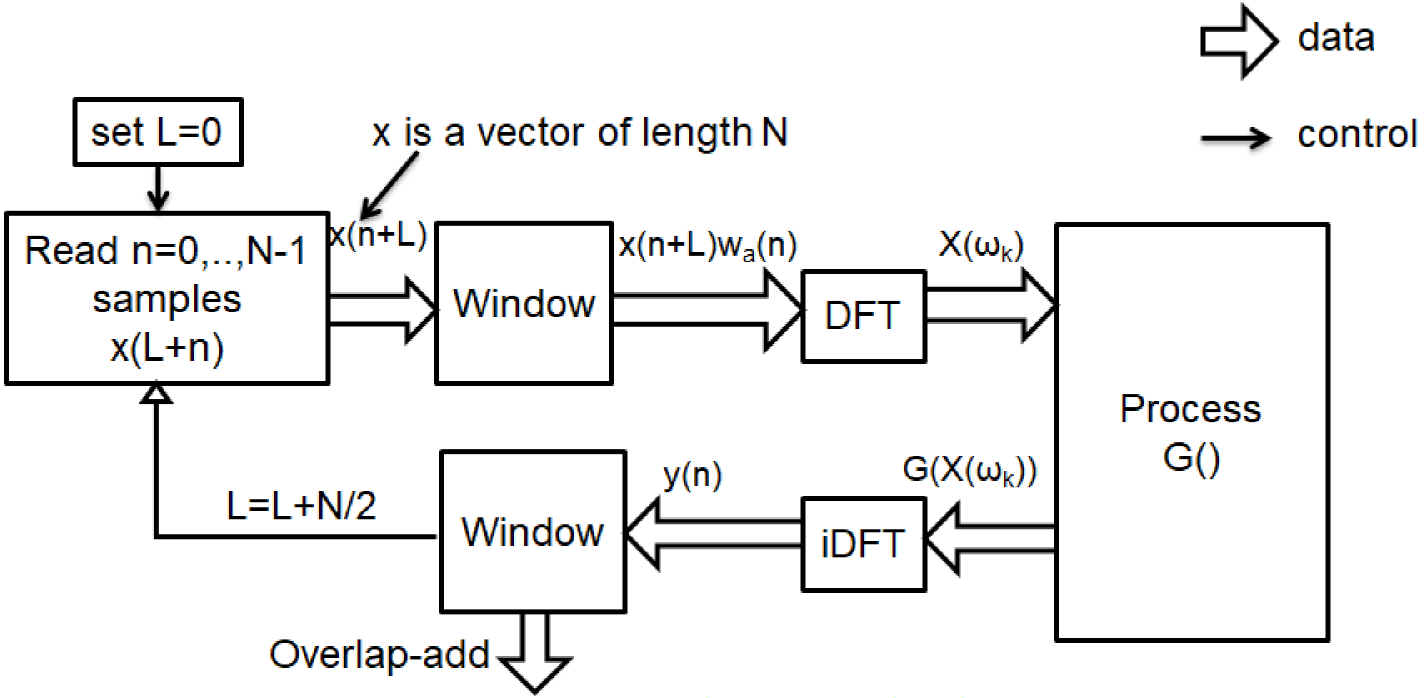
Overlap-and-add processing

- Reconstruction of the signal done using inverse trasform IDFT
- Analysis part is not concerned about reconstruction:
  - Taking the inverse DFT of the STFT frames that do not overlap in time, the ends of the adjacent frames will not connect well
  - Results in audible “ripple” distortion
- Solution: have frames overlap

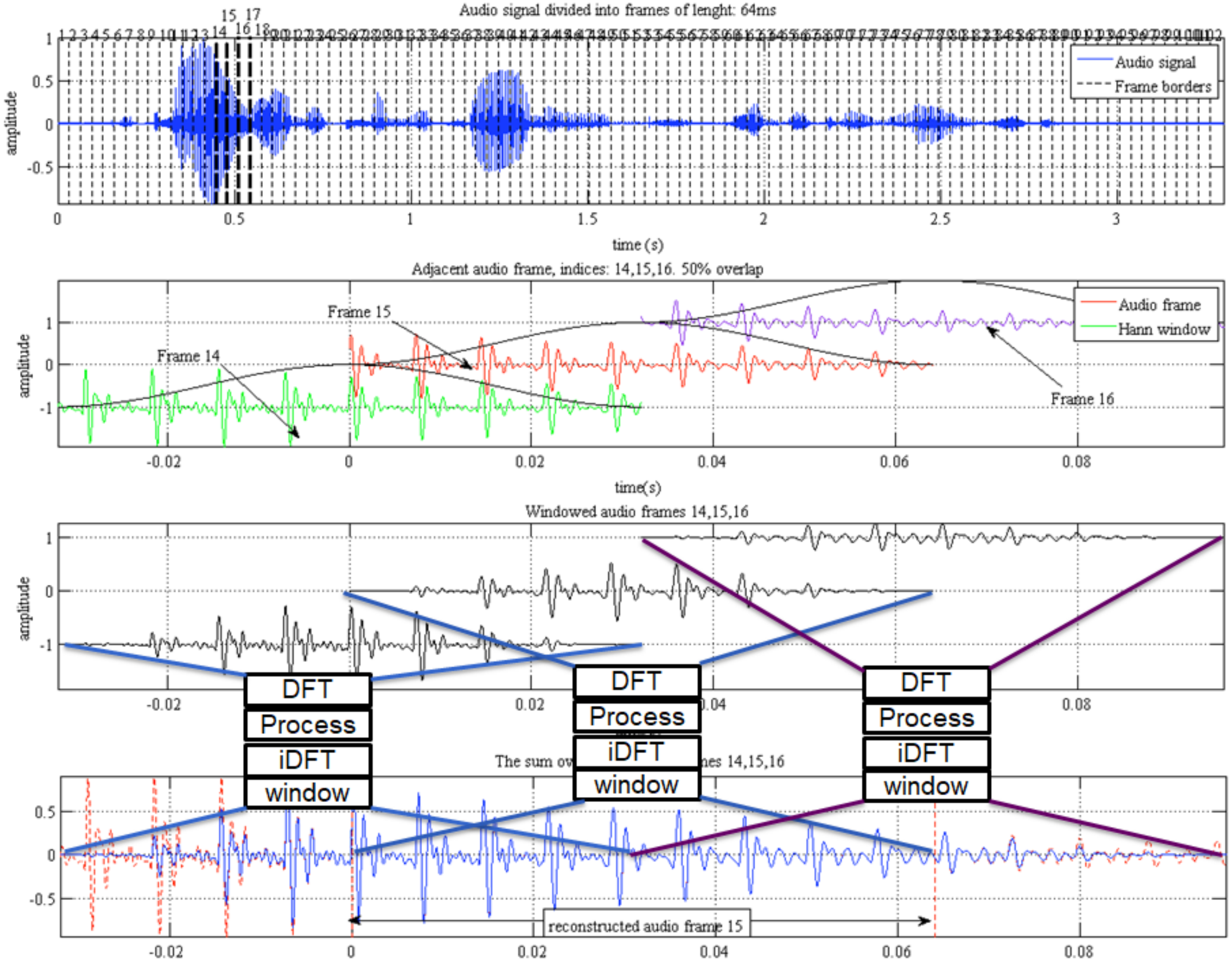
Window overlap-add processing:

- Move the analysis window (e.g.) 50% of the window length forward
- Apply windowing
- Take DFT
- Process data (in frequency domain)
- Take iDFT
- Apply windowing (the second time).
- Reconstruct a frame by summing with previous frame with overlap

STFT and overlap-and-add processing



$r(L+n)=r(L+n)+y(n)w_s(n)$, *r* is the reconstructed signal



Audio signal divided into frames of lenght: 64ms
Audio signal
Frame borders
amplitude
time (s)
Adjacent audio frame, indices: 14,15,16. 50% overlap
Frame 14
Frame 15
Frame 16
Audio frame
Hann window
time(s)
Windowed audio frames 14,15,16
DFT
Process
iDFT
window
DFT
Process
iDFT
window
DFT
Process
iDFT
window
The sum ov
nes 14,15,16
reconstructed audio frame 15

Overlap-and-add processing

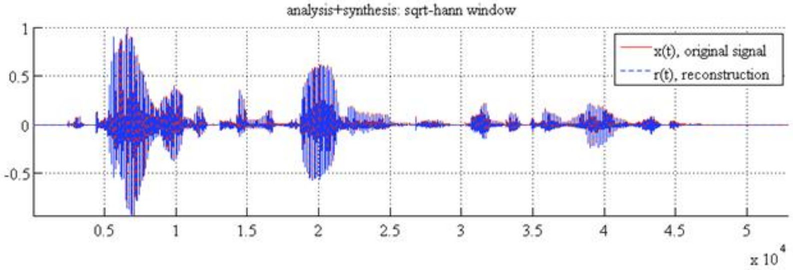
- Note that the data is weighted twice
  - Once in the analysis before DFT using wa(n)
  - After iDFT and before reconstruction using ws(n)
- The condition for perfect reconstruction (50% overlap) is

$$w_a(n) \cdot w_s(n) + w_a(n + N/2) \cdot w_s(n + N/2) = 1$$

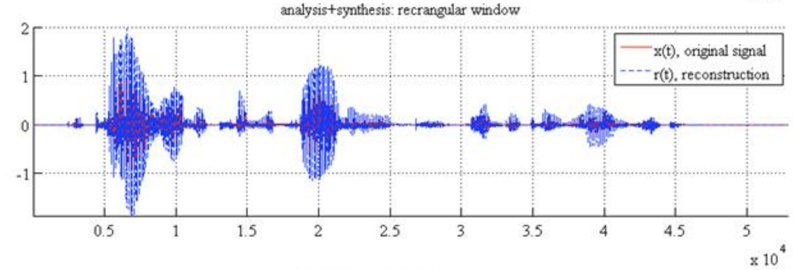
where $w_a(n)$ and $w_s(n)$ are analysis and reconstruction windows, respectively with window length N samples

- Often the “square root” of a window function (such as hann) is taken to be used in analysis and reconstruction
  - However, in general the windows don’t need to be symmetric
- STFT processing can be considered to satisfy perfect reconstruction (if window criteria above is met)

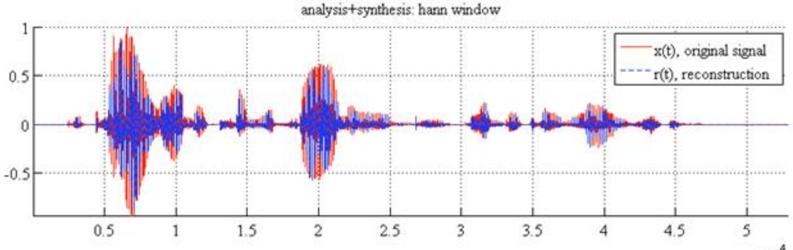
Overlap-add reconstructions



sqrt-hann in both analysis and synthesis
i.e. sin-window

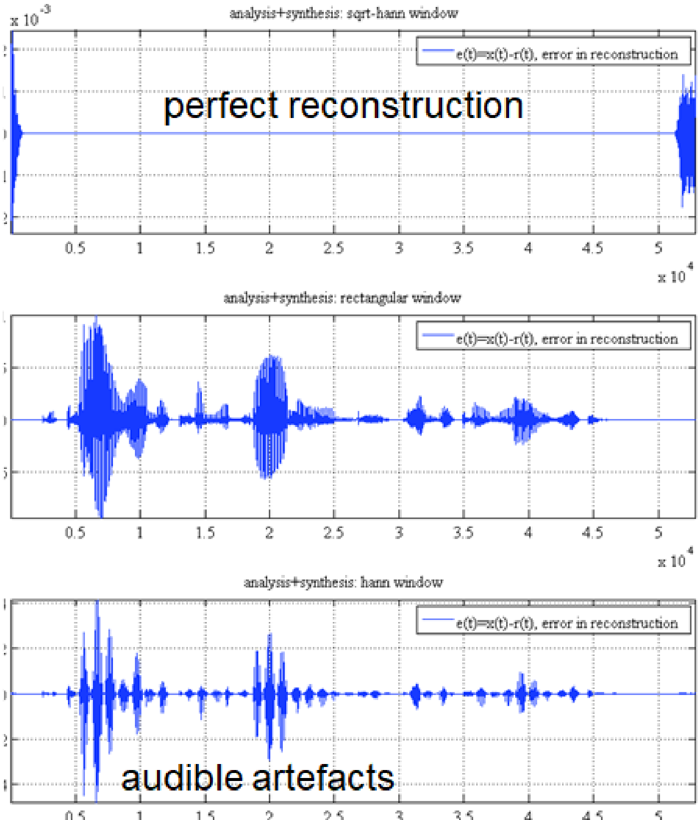


rectangular in both analysis and synthesis
i.e. sin-window



hann in both analysis and synthesis

Reconstruction error



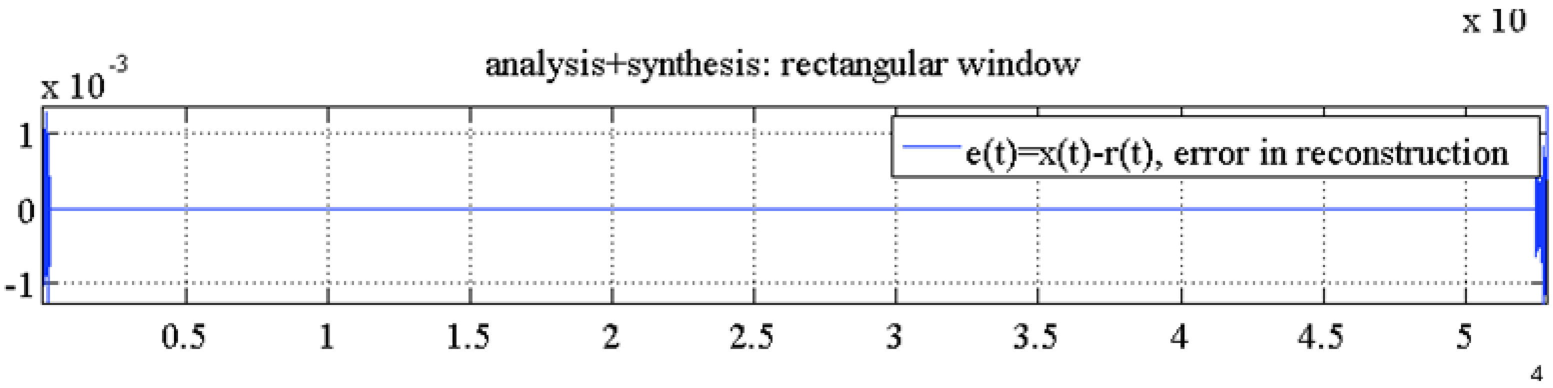
sqrt-hann in both analysis and synthesis
first+last frame see zero-padding

rectangular in both analysis and synthesis
i.e. sin-window

hann in both analysis and synthesis

Rectangular windowing

- Since $w_a(n)*w_s(n)+w_a(n+N/2)*w_s(n+N/2)=1*1+1*1=2$, the reconstructed signal's amplitude is twice the original signal's amplitude. (see figs on previous pages)
- The reconstructed signal should be scaled with 0.5 to enable perfect reconstruction.
- With this scaling, the reconstruction error is 0 (perfect reconstruction)
- However, the spectral smearing is still present



Note on implementation

Some Python implementation such as scipy.signal.stft and librosa.stft do not use sqrt(window), but instead estimate the sum of squared overlapping windows

$$w_s(n) = \frac{w_a(n)}{w_a^2(n)+w_a^2\left(n+\frac{N}{2}\right)}$$

This sum signal is used to normalize the reconstructed output signal.

STFT

```
while (ei<len(x)):

    wi=wi+1;
    x_win = x[si:ei]*win
    X = np.fft.rfft(x_win,winlen) if wi==0 \
        else np.vstack((X,np.fft.rfft(x_win)))

    if wi < 10:
        print si,ei,np.shape(X)

    si=si+winlen/2
    ei=si+winlen
```

iSTFT

```
while (ei<len(x)):
    wi=wi+1;
    r = np.fft.irfft(X[wi,:])
    rec[si:ei] = rec[si:ei] + r * win

    ifft_window_sum[si:ei] += win**2

    si=si+winlen/2
    ei=si+winlen

rec /= ifft_window_sum
```

Scaling with squared window

vs

STFT

```
while (ei<len(x)):

    wi=wi+1;
    x_win = x[si:ei]*np.sqrt(win)
    X = np.fft.rfft(x_win,winlen) if wi==0 \
        else np.vstack((X,np.fft.rfft(x_win)))

    if wi < 10:
        print si,ei,np.shape(X)

    si=si+winlen/2
    ei=si+winlen
```

iSTFT

```
while (ei<len(x)):
    wi=wi+1;
    r = np.fft.irfft(X[wi,:])
    rec[si:ei] = rec[si:ei] + r * np.sqrt(win)

    si=si+winlen/2
    ei=si+winlen
```

Square-root window applied

Thinking break (2 minutes)

Discrete Cosine Transform (DCT)

- Expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies
  - Similar to DFT (but DFT uses sines and cosines)
- Fewer cosine functions are needed to approximate a typical signal
  - Makes it very useful in signal compression

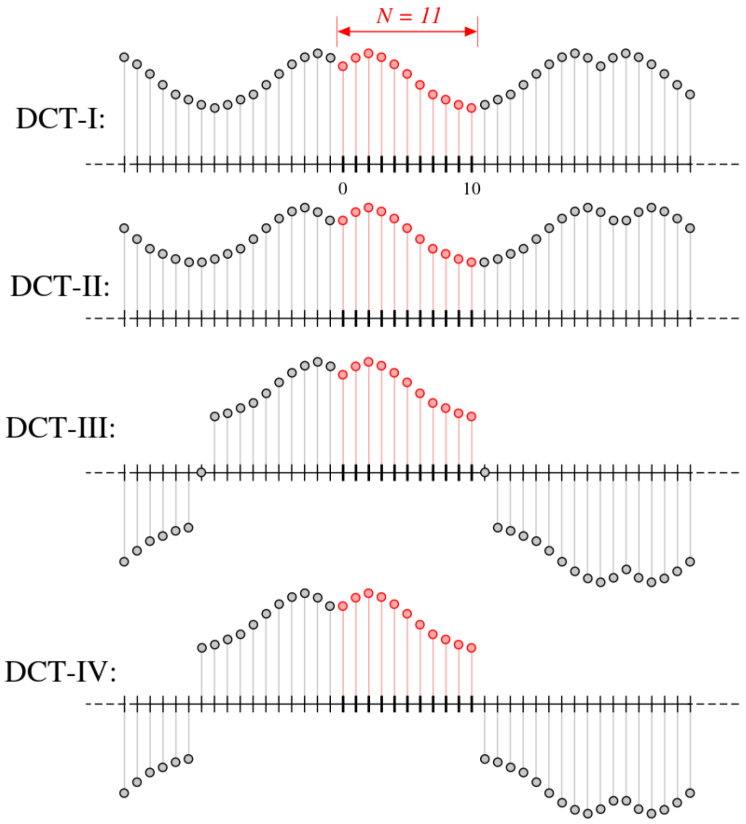
- DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (Fourier transform of a real and even function is real and even)

- Four variants common (eight in total)
  - DCT II - used in visual media standards (image compression JPEG, video compression MPEG, etc)
  - DCT IV or MDCT used in most audio standards ,both general audio and speech (MP3, AAC, WMA, Vorbis, etc)

DCT variants: domain boundaries

- Fourier-related transforms: operate on a function over a finite domain
  - define an extension of that function outside the domain
  - DFT: periodic extension of the original function
  - DCT: even extension of the original function.

- Different boundary conditions affect this extension
  - This strongly affects the applications of the transform
  - Also determines the different variants of the DCT



Modified discrete cosine transform MDCT

- Lapped transform based on type-IV DCT
- Subsequent blocks are overlapped so that the last half of one block coincides with the first half of the next block → helps to avoid artifacts stemming from the block boundaries
- Has half as many outputs as inputs
- 2N real numbers x0, ..., x2N-1 are transformed into the N real numbers X0, ..., XN-1

- Analysis with **half-overlapping frames**:
  - frame input signal x(n), n=0,…,2N-1
  - length 2N
  - hop-size N samples

MDCT definition

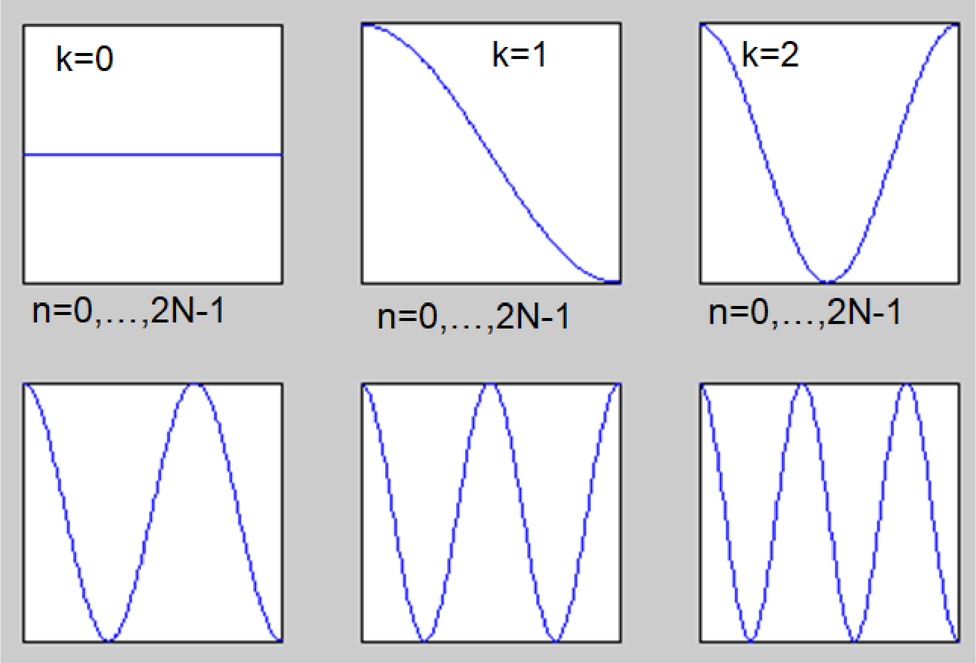
MDCT: $$X(k)=\sum_{n=0}^{2N-1} w_a(n)x(n)\cos\left(\frac{\pi}{N}\left(n+\frac{1+N}{2}\right)\left(k+\frac{1}{2}\right)\right), k=0,\ldots,N-1$$

iMDCT: $$y(n)=w_s(n)\frac{2}{N}\sum_{k=0}^{N-1} X(k)\cos\left(\frac{\pi}{N}\left(n+\frac{1+N}{2}\right)\left(k+\frac{1}{2}\right)\right)$$

$$w_a(n)=w_s(n)=\sin\left(\frac{\pi}{4N}(2n+1)\right),\quad \text{(also other windowing pairs exist)}$$

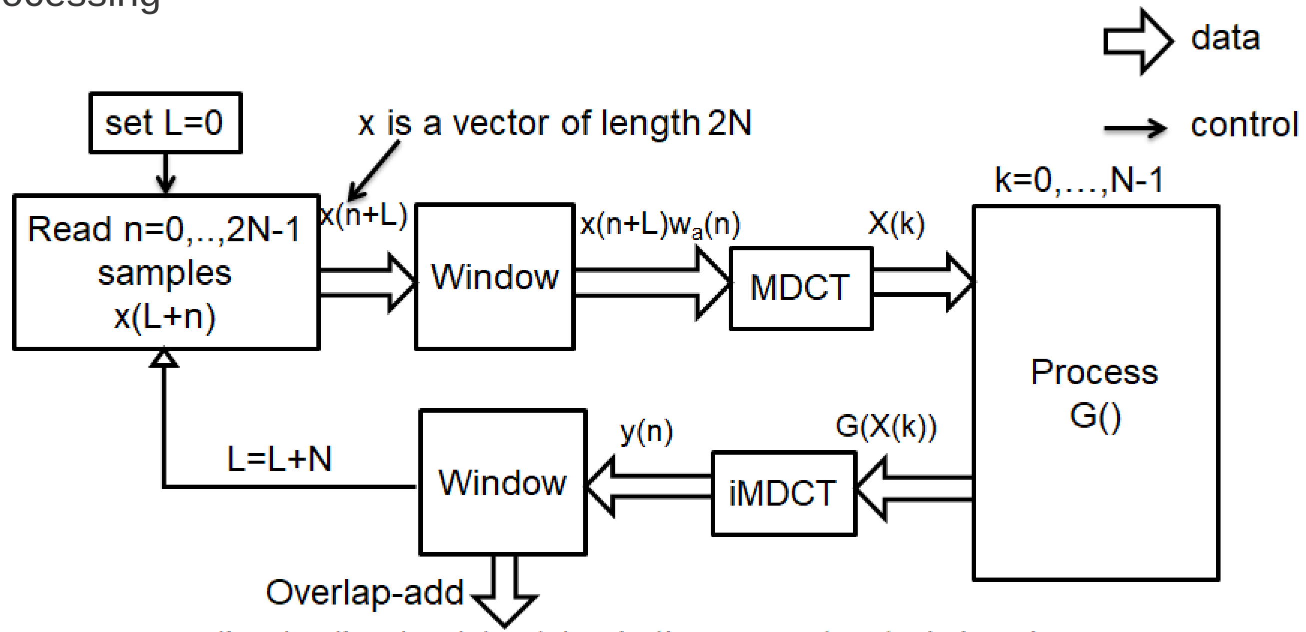
Discrete Cosine Transform

First few basis functions (real-valued)



Window overlap-and-add in MDCT

MDCT processing



$r(L+n)=r(L+n)+y(n)w_s(n)$, r is the reconstructed signal

Thinking break (2 minutes)

Filterbanks

Introduction

- Filterbanks transform the broadband time-domain input signal into narrowband time-domain signals using **band-pass filters.**
- Filterbanks are used for example in:
  - Perceptual audio coding
  - Multi-band equalizers
  - Bandwise dynamic range control
  - Machine hearing and audio content analysis
  - Signal enhancement

- The human auditory system performs frequency analysis
  - Critical bands in hearing, structure of the inner ear
  - One reason why we encounter filterbanks in many audio processing applications

Tampere University

Filterbanks in audio coding

In audio coding, the input signal is processed at subbands

- A filterbank is required, in other words, a set of filters that select neighbouring narrow subbands that cover the entire frequency range
- Audio coding (separate lecture) removes inaudible components of audio.

The filterbanks used in audio coding consist of:

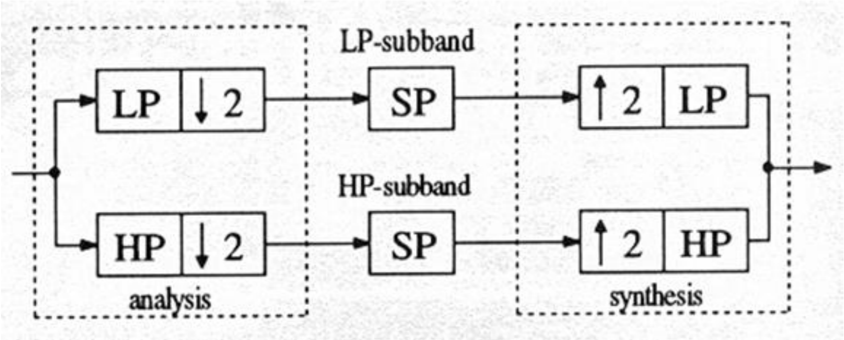
- Analysis filterbank that decomposes a signal into subbands
- Synthesis filterbank that reconstructs a wideband signal to the output

Critically sampled, perfect reconstruction filterbanks

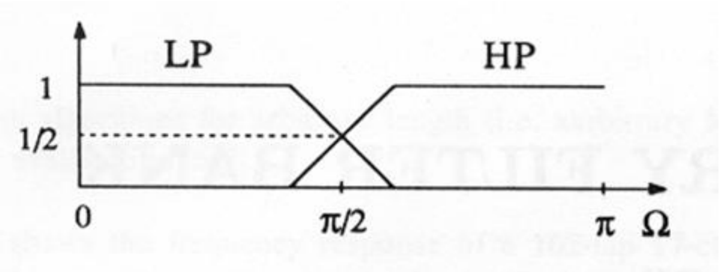
- **Critical sampling**: if the filterbank subdivides the frequency range into K bands, the signal at each band is downsampled by factor 1/K
    - Amount of data does not increase
- **Perfect reconstruction:** if no processing takes place at subbands, the signal can be reconstructed without errors using a synthesis filterbank

Critical sampling at two sub-bands

Two-band critically sampled analysis-synthesis filterbank



Magnitude responses of the filters applied at the two bands

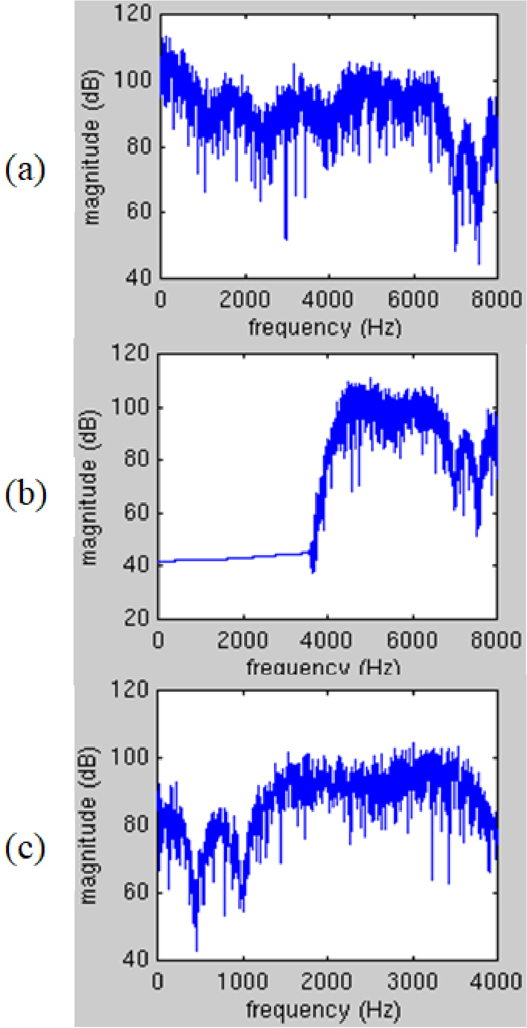


Analysis filterbank

- What happens in the analysis filterbank?
  - LP + 2 : Lowpass filter and downsample by factor 2
  - HP + 2 : Highpass filter and downsample by factor 2

- When the upper half-band [fs/4 fs/2] is decimated, it is aliased (mirrored) to the lower frequencies [0 fs/4]
  - The aliasing does not corrupt spectral information since the lower frequency components were filtered out using a highpass filter

- Upper half-band:
  - (a) original signal spectrum,
  - (b) highpass filtered (HP) spectrum
  - (c) highpassed and decimated (HP + 2), aliased spectrum Note the lower sampling rate in panel c): Nyquist freq.is 4 kHz



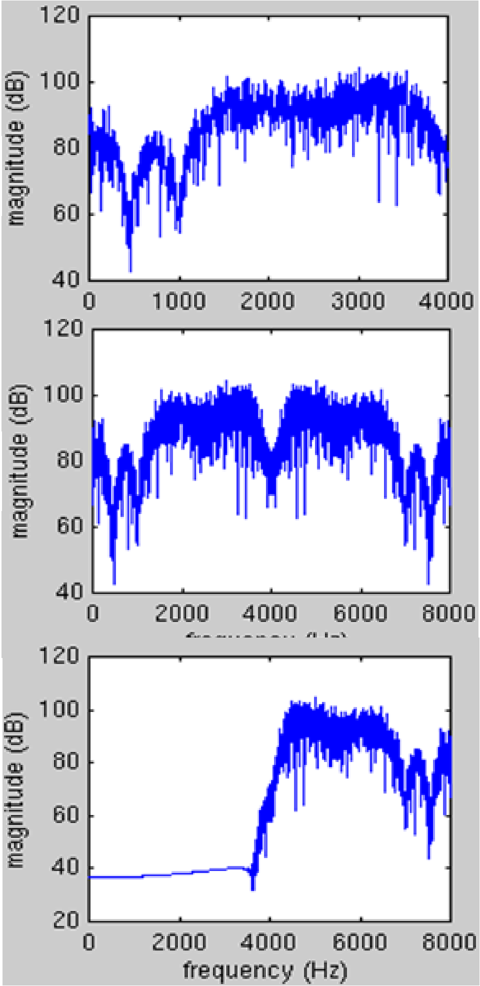
Synthesis filterbank

- What happens in the synthesis filterbank?
  - ↑2 + LP : Upsample by factor 2 and lowpass filter
  - ↑2 + HP : Upsample by factor 2 and highpass filter

- ↑2 operation in practice:
  - Add zeros between the sample values in the signal (vector of numbers)
  - Multiply the signal by 2 in order to keep its level unchanged

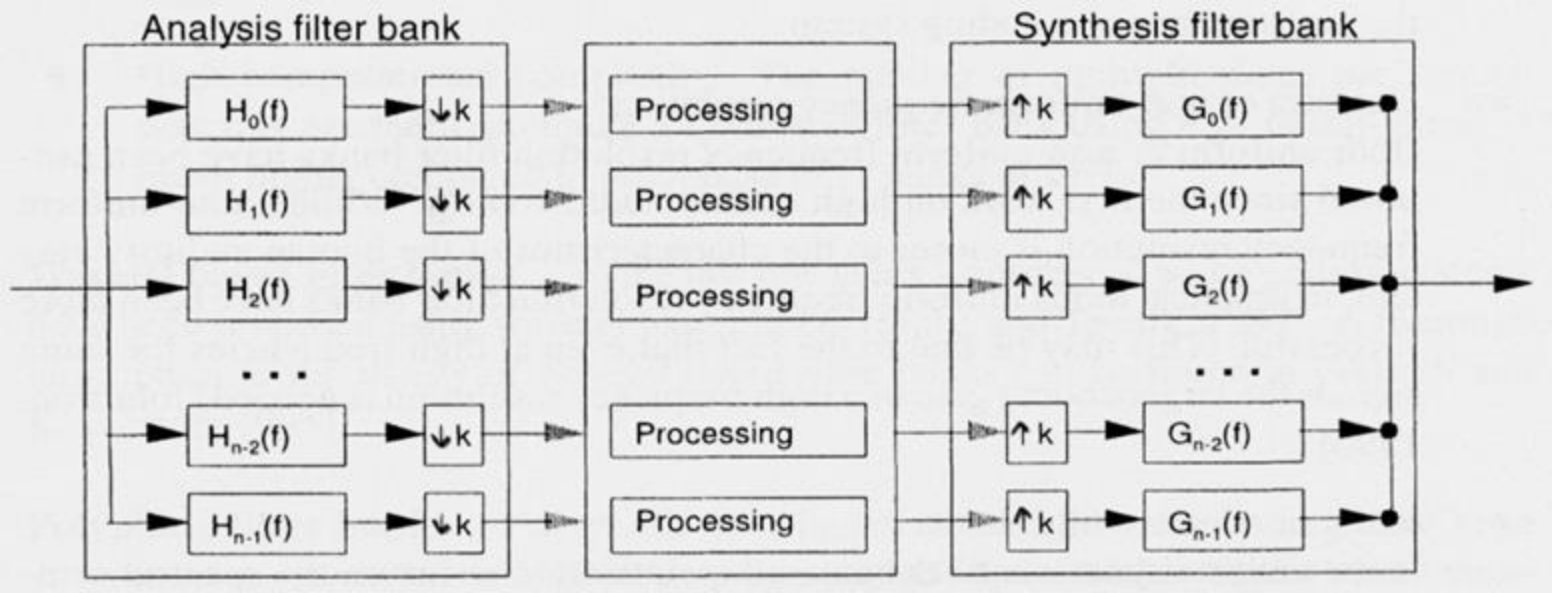
- Upper half-band:
  - (c) spectrum of the signal that was highpassed and decimated in analysis bank
  - (d) spectrum obtained by interpolating (↑2) signal in c
  - (e) after interpolating and highpass filtering (↑2 + HP) the signal in c

(c)
(d)
(e)



Multiple bands

- The principle scales to multiple bands, each subband decimated by factor *k*
- Critical sampling if *n*=*k*



Multiple bands

- What happens in the analysis bank at *n* subbands?
    - Spectrum in range [0, $f_s/2$] is divided into n bands, each of width ($f_s/2$) / *n*
    - Bandpass filter $H_m(f)$ in the analysis bank selects band *m*
    - Band *m* covers the frequencies

$$\left[\frac{mf_s}{2n}, \frac{(m+1)f_s}{2n}\right] \quad , m = 0, 1, \ldots, n-1$$

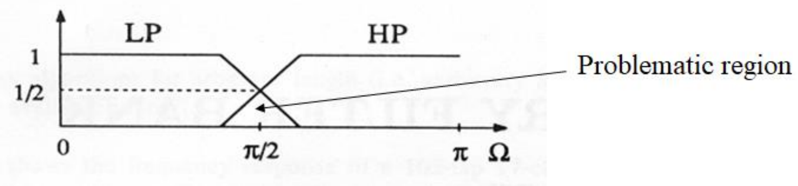
    - In downsampling, the band is aliased to frequencies $\left[0, f_s/(2n)\right]$
        - No problem, since those frequencies were filtered out by $H_m(f)$

- In the synthesis bank
    - Interpolation by factor *k* (*k*=*n*) replicates the subband $\left[0, f_s/(2n)\right]$ at all subbands
    - Each subband is selected at its correct frequency range using synthesis bandpass filter $G_m(f)$ (same passband as $H_m(f)$)

Tampere University

Aliasing error

- In a critically-sampled filterbank some unwanted aliasing happens at the subbands
  - Filters are not ideal (transition band, not step function)
- For example when downsampling by factor 2, the part that exceeds the new Nyquist frequency /2 (= fs/4) is aliased

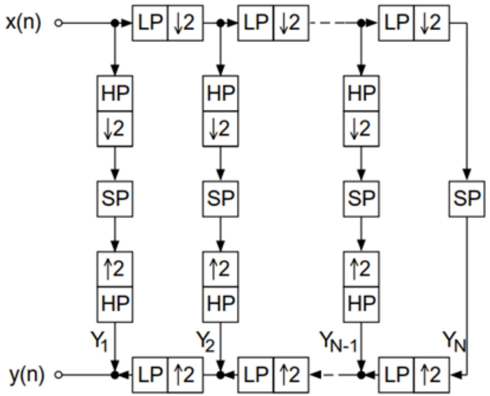


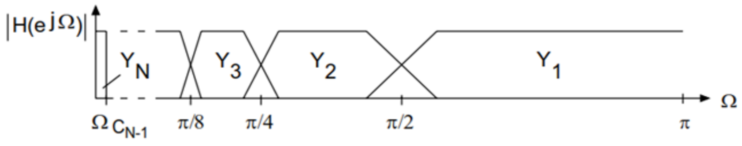
- The filterbanks used in audio coding are usually designed so that the synthesis bank eliminates the aliasing that occurs at the subbands
  - Achieves perfect or near-perfect reconstruction despite the unwanted aliasing at subbands
  - QMF = Quadrature Mirror Filter bank: designed so that the aliasing that happens in analysis part is eliminated in the synthesis part.

Multiple-band QMF

Cascaded QMF structure to create *M* bands:

- Uniform band analysis:
    - The high-pass and low-pass bands are both divided again into two bands to receive 4 equally spaced bands. The process can be repeated to obtain 8 bands, 16 bands, etc.
- Octave band analysis filter-bank
    - Can be created by splitting the lower band of the two-band QMF into two bands and keeping the higher band as is. The process is repeated until a sufficient band-spacing is obtained.





Filterbanks vs transforms

- In a filterbank, the signal at subband *k* is obtained by convolving the filter $h_k(n)$ with the input signal, computed every *M* samples (downsampling)
- In a transform, the coefficient corresponding to basis function *k* is obtained as an inner product between the windowed signal and basis vector $g_k(n)$
    - STFT uses complex basis vector
    - MDCT uses cosine basis vectors
- Differences in implementation: transforms are fast to compute when there are a lot of subbands
- Filterbank implementation facilitates non-uniform frequency resolution and specification of the filters separately for each band

Summary

- Frequency transform is often applied in modern audio processing applications
- Machine learning applications often apply STFT
- Modern audio codecs use (MDCT)
  - E.g. Dolby AC-2,AC-3, MPEG-2 AAC
- Steps required in transform processing: overlap processing, windowing, reconstruction

- What does a filter bank do, and where it is used
- Design criteria and properties of filter-banks

- Transforms and filter banks are basically two views of the same thing