Audio signal processing

Erik Pärlstrand

May 2018

1 Introduction to discrete Fourier transforms

Real sinusoid

A real sinusoid in a discrete time domain can be expressed by:

$$x[n] = Acos(2\pi fnT + \phi)$$

where x is the array of real values of the sinusoid, n is an integer value expressing the time index, A is the amplitude value of the sinusoid, f is the frequency value of the sinusoid in Hz, T is the sampling period equal to $1/f_s$, f_s is the sampling frequency in Hz, and ϕ is the initial phase of the sinusoid in radians.

Complex sinusoid

A complex sinusoid in a discrete time domain can be expressed by:

$$\bar{x}[n] = Ae^{j(\omega nT+\phi)} = Acos(\omega nT + \phi) + jAsin(\omega nT + \phi)$$

where $\bar{x}$ is the array of complex values of the sinusoid, n is an integer value expressing the time index, A is the amplitude value of the sinusoid, ω is the frequency of the sinusoid in radians per second (equal to $2\pi f$), T is the sampling period equal $1/f_s$, f_s is the sampling frequency in Hz and ϕ is the initial phase of the sinusoid in radians.

Discrete Fourier Transform (DFT)

The N point DFT of a sequence of real values x (e.g. a sound) can be expressed by:

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

where n is an integer value expressing the discrete time index, k is an integer value expressing the discrete frequency index, and N is the length of the DFT. One can view the DFT as correlating the sinusoid ($e^{-j2\pi kn/N}$) with the signal $x[n]$.

Inverse Discrete Fourier Transform (IDFT)

The IDFT of a spectrum X of length N can be expressed by:

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

where $n = 0, ..., N-1$. Here, n is an integer value expressing the discrete time index, k is an integer value expressing the discrete frequency index, and N is the length of the spectrum X.

Fast Fourier Transform (FFT)

An efficient way to compute the discrete Fourier transform of a signal is the fast Fourier transform. Normal DFT has a time complexity of $\mathrm{O}(n^2)$ while FFT has a complexity of $\mathrm{O}(n \log n)$.

The FFT algorithm factorizes the DFT matrix in order to exploit the symmetries in the DFT equation. FFT computation is specially very efficient when the FFT size is a power of 2. Therefore, whenever possible we use an FFT size that is a power of 2.

Magnitude spectrum

The magnitude of a complex spectrum X is obtained by taking its absolute value: $|X[k]|$. Sometimes the magnitude is expressed in a decibel scale: $20 \cdot log_{10}|X[k]|)$.

Energy of a signal

The energy of a signal $x[n]$ of length N can be computed in the discrete time domain as follows:

$$E = \sum_{n=0}^{N-1} |x[n]|^2$$

Energy in a frequency band

Given the DFT spectrum of the signal $X[k]$, the energy E in a specific frequency band spanning the bin index k_1 to k_2 can be computed as:

$$E = \sum_{k=k_1}^{k_2} |X[k]|^2$$

Note that in this computation the $X[k]$ values are not in decibels.

Signal to noise ratio (SNR)

Signal to noise ratio (SNR) is a frequently used measure to quantify the amount of noise present/added in a signal. It can be computed as:

$$\mathrm{SNR} = 10 \log_{10} \left(\frac{E_{\text{signal}}}{E_{\text{noise}}} \right)$$

where, E_{signal} and E_{noise} is the energy of the signal and the noise respectively.

2 Fourier Properties

Convolution and filtering

The convolution operation between two series x_1 and x_2 is defined as:

$$(x_1 * x_2)[n] = \sum_{m=-\infty}^{\infty} x_1[m]x_2[n-m]$$

In signal processing, convolution is a mathematical operation that can be used for filtering. Filtering involves selectively suppressing certain frequencies present in the signal. Filtering is often performed in the time domain by the convolution of the input signal with the impulse response of a filter. However, the DFT has the following property:

Convolution in the time domain results in multiplication in frequency domain, i.e. $x_1[n] * x_2[n] \implies X_1[k]X_2[2]$.

Therefore, the same operation can also be done in the DFT domain using this properties, by multiplying the DFT of the input signal by the DFT of the impulse response of the filter.

Let's consider an example of filtering, namely linear filtering:

An example of such a case is if a signal f contains some noise and one wants to generate an average of f over a certain period in time. One can then choose a signal g of an appropriate shape and let it slide over f. The resulting signal h then illustrates an average of f during a selected number of time units. This gives a clearer signal where insignificant deviations have been removed.

Zero-padding

Zero-padding a signal is done by adding zeros at the end of the signal. If we perform zero-padding to a signal before computing its DFT, the resulting spectrum will be an interpolated version of the spectrum of the original signal. In most implementations of the DFT (including the FFT algorithms) when the DFT size is larger than the length of the signal, zero-padding is implicitly done.

Zero phase windowing

Zero phase windowing of a frame of signal puts the centre of the signal at the "zero" time index for DFT computation. By moving the centre of the frame to zero index by a circular shift, the computed DFT will not have the phase offset which would have otherwise been introduced. This is caused by the following property of the DFT:

A shift causes the DFT to be multiplied by a complex exponential, i.e.

$$x[n-n_0] = e^{-j2\pi k n_0/N} X[k]$$

When used in conjunction with zero-padding, zero phase windowing is also useful for the creation of a frame of length of power of 2 for FFT computation.

3 STFT and window functions

Window function

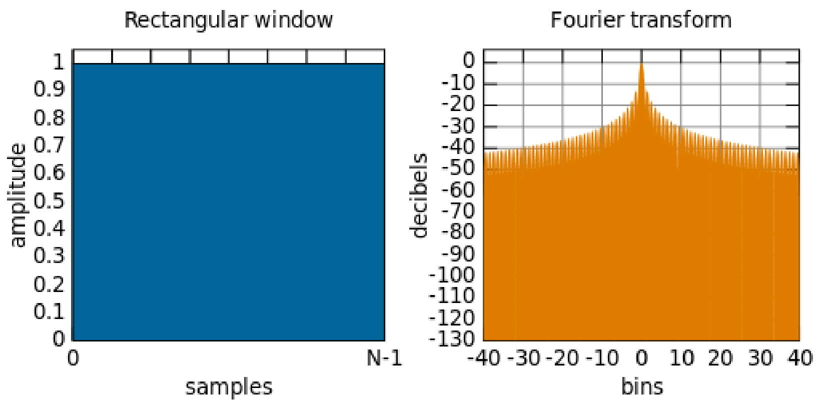
A window function is a function that is zero-valued outside of some chosen interval. When a data-sequence is multiplied by a window function, the product is also zero-valued outside the interval: all that is left is the part where they overlap.

The window function are symmetric. Now, let M represents the width (in samples) of the window function and $0 \le n \le M-1$. Below are some examples of window functions:

Rectangular window: It's defined by:

$$w[n] = 1$$

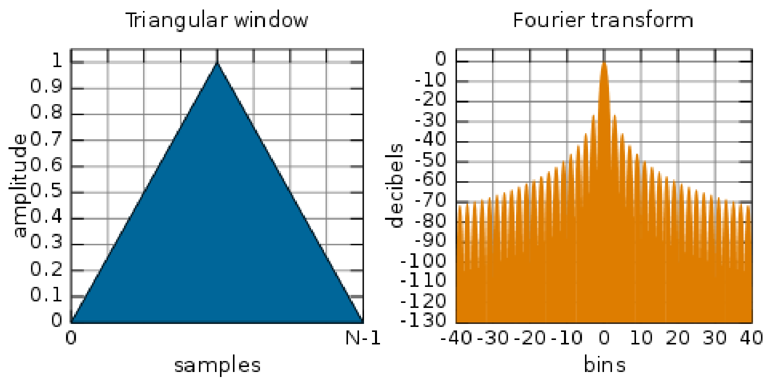
Below is an illustration of the window function and its Fourier transform:



Triangular window: It's defined by:

$$w[n] = 1 - \left| \frac{n - \frac{N-1}{2}}{\frac{L}{2}} \right|$$

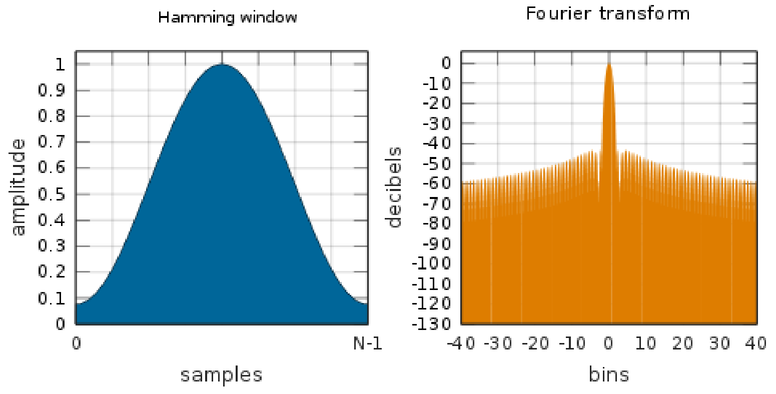
where L can be N, $N+1$ or $N-1$. Below is an illustration of the window function and its Fourier transform:



Hamming window: It's defined by:

$$w[n] = 0.54 - 0.46\cos\big(\frac{2\pi n}{M-1}\big)$$

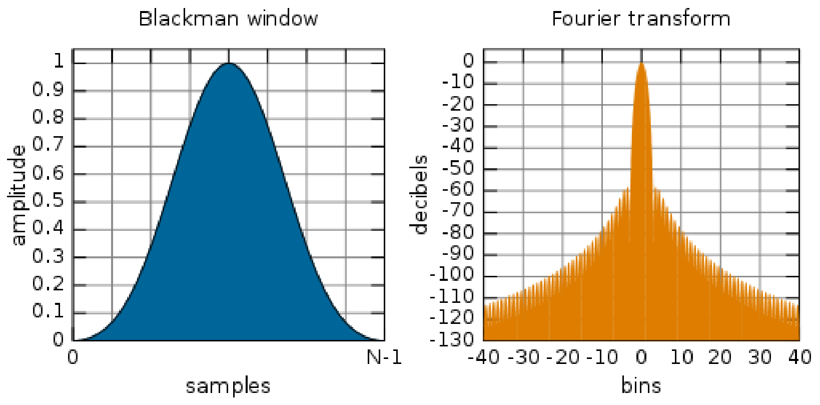
Below is an illustration of the window function and its Fourier transform:



Blackman window: It's defined by:

$$0.42 + 0.5\cos\big(\frac{2\pi n}{M-1}\big) + 0.08\cos\big(\frac{4\pi n}{M-1}\big)$$

Below is an illustration of the window function and its Fourier transform:



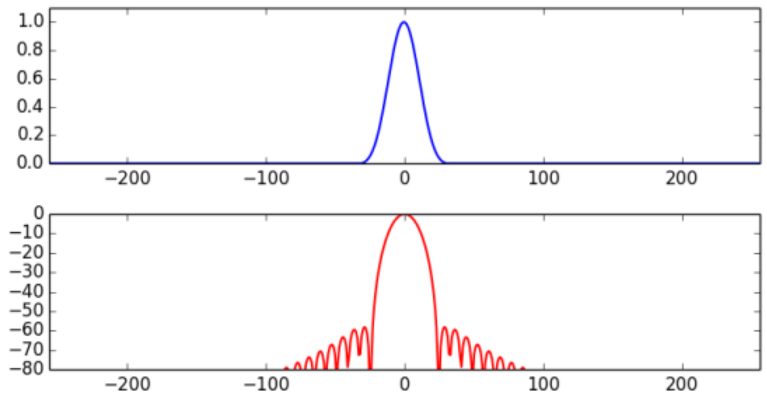
The DFT of the window is called filter in the frequency domain. Smoothing the signal in the time domain correspond to low pass filtering in the frequency domain.

Main lobe of the spectrum of a window

The width of the main lobe of the magnitude spectrum of a window is an important characteristic used in deciding which window type is best for the analysis of an audio excerpt. There exists a trade-off between the main lobe width and the side lobe attenuation.

Typically for windows with a narrower main lobe, the side lobes are less attenuated. An example of the Blackman window is given below. The figure shows the plot of the time domain window (blue) together with its magnitude spectrum (red). Both the time domain function and its spectrum are centered around 0.

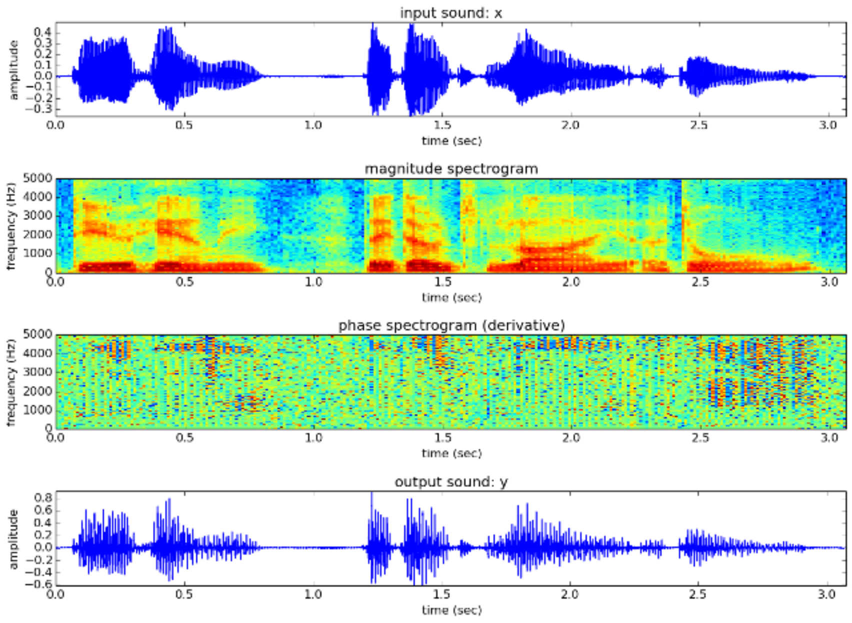
An interesting fact is that changing the length of a window M doesn't affect the main lobe width of the spectrum of the window in samples. Note that if you use zero-padding for computing the spectrum of a window, the main lobe width will be multiplied by the zero-padding factor.



Short-Time Fourier Transform (STFT)

The short-time Fourier transform is a Fourier-related transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time.

In practice, the procedure for computing STFTs is to divide a longer time signal into shorter segments of equal length and then compute the Fourier transform separately on each shorter segment. This reveals the Fourier spectrum on each shorter segment. One then usually plots the changing spectra as a function of time. Below is an example of the STFT:



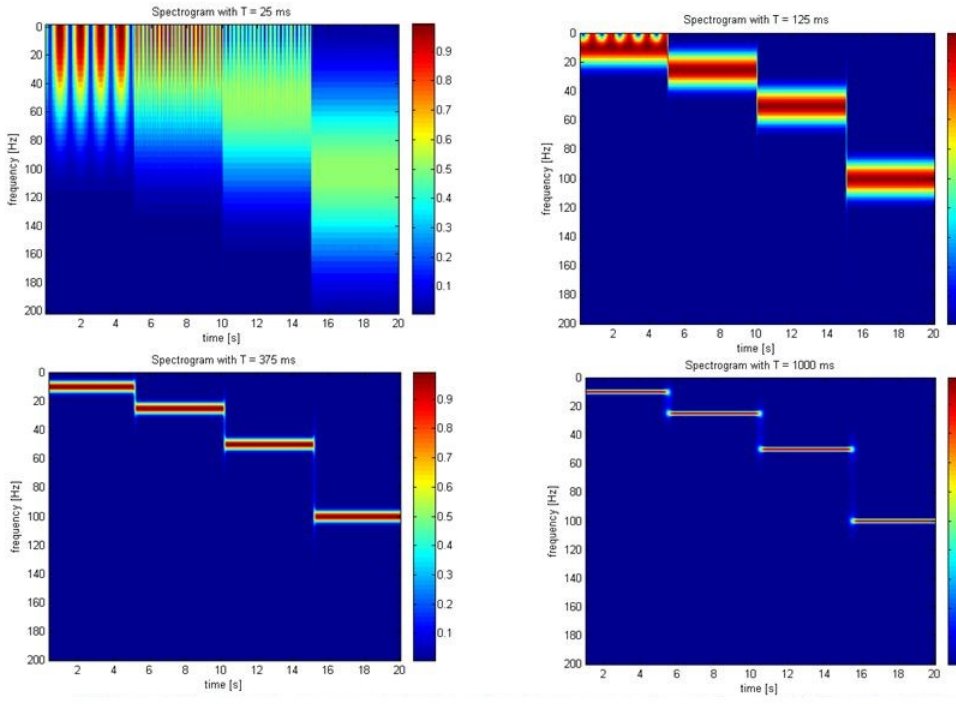
In discrete time, the data to be transformed could be broken up into chunks or frames (which usually overlap each other, to reduce artifacts at the boundary). Each chunk is Fourier transformed, and the result is added to a matrix, which records magnitude and phase for each point in time and frequency. This can be expressed as:

$$X_l[k] = \sum_{n=-N/2}^{N/2-1} w[n]x[n + lH]e^{-j2\pi kn/N}$$

where w is the analysis window, l is the frame number and H is the hop-size.

One problem with the STFT is that it has a fixed resolution. The width of the windowing function relates to how the signal is represented; it determines whether there is good frequency resolution (frequency components close together can be separated) or good time resolution (the time at which frequencies change).

A wide window gives better frequency resolution but poor time resolution. A narrower window gives good time resolution but poor frequency resolution. Below is an illustration about this issue:



4 Mel-scale filter banks

Mel-scale filter banks are a popular pre-processing step used in speech processing for machine learning (deep learning).

In a nutshell, a signal goes through a pre-emphasis filter; then gets sliced into (overlapping) frames and a window function is applied to each frame; afterwards, we do a DFT on each frame and calculate the power spectrum; and subsequently compute the filter banks. A final step is mean normalization.

Pre-emphasis filter

The first step is to apply a pre-emphasis filter on the signal to amplify the high frequencies. A pre-emphasis filter is useful in several ways: (1) balance the frequency spectrum since high frequencies usually have smaller magnitudes compared to lower frequencies, (2) avoid numerical problems during the DFT operation and (3) may also improve the SNR.

The pre-emphasis filter can be applied to a signal x using the first order filter in the following equation:

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

where typical values for the filter coefficient α are 0.95 or 0.97

Fourier-transform and Power Spectrum

After pre-emphasis, we will apply STFT. One usually use the Hamming window (described above).

Furthermore, typical frame sizes in speech processing range from 20 ms to 40 ms with 50% (±10%) overlap between consecutive frames. Popular settings are 25 ms for the frame size and a 10 ms stride (15 ms overlap). Moreover, one usually chooses N to be 256 or 512. The power spectrogram (periodogram) is given by:

$$P = \frac{|X_l[k]|^2}{N}$$

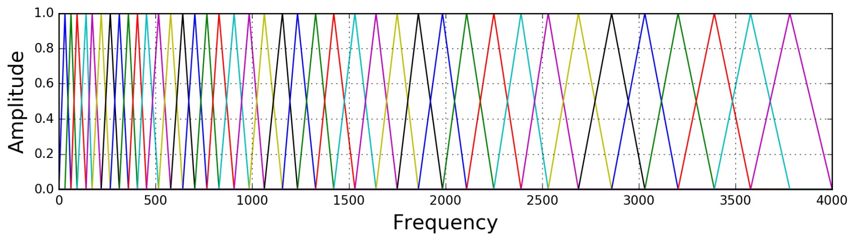
Mel-scaled filter banks

The final step to computing filter banks is applying triangular filters, typically 40 filters, on a Mel-scale to the power spectrum to extract frequency bands. The Mel-scale aims to mimic the non-linear human ear perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies. We can convert between Hertz (f) and Mel (m) using the following equations:

$$M = 2596 \log_{10}\big(1 + \frac{f}{700}\big)$$

$$f = 700(10^{m/2596} - 1)$$

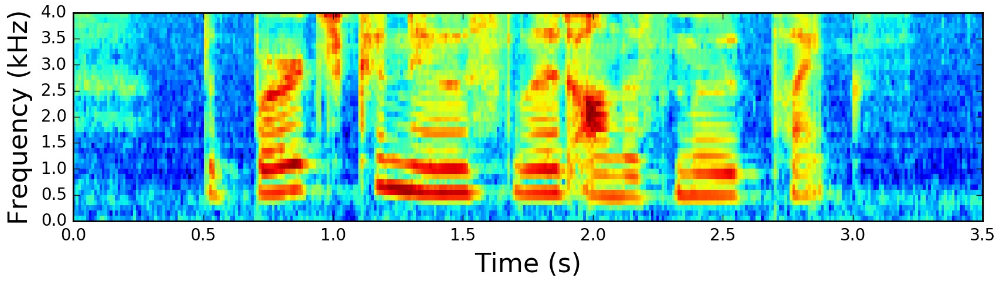
Each filter in the filter bank is triangular having a response of 1 at the center frequency and decrease linearly towards 0 till it reaches the center frequencies of the two adjacent filters where the response is 0, as shown in this figure (in Mel-scale):



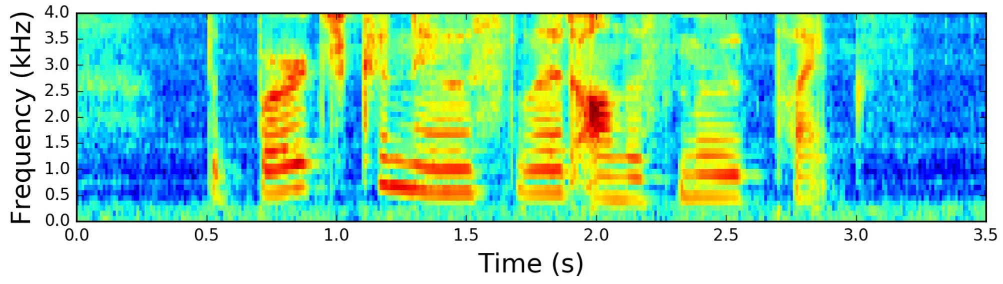
This can be modeled by the following equation:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leq k < f(m) \\ 1 & k = f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) < k \leq f(m+1) \\ 0 & k > f(m-1) \end{cases}$$

After applying the filter bank to the periodogram of the signal, we obtain the following spectrogram:



To balance the spectrum and improve the SNR, we can subtract the mean of each coefficient from all frames. Doing yields:


Frequency (kHz)
Time (s)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0