Carnegie Mellon ADSP Electrical & Computer ENGINEERING
Advanced Digital Signal Processing (18-792)
Fall Semester 2019
Department of Electrical and Computer Engineering

NOTES ON SHORT-TIME FOURIER TRANSFORMS

I. Introduction

While frequency-domain representations such as the DTFT and the DFT are useful, they both are obtained by summing the time function $x[n]$ from $-\infty$ to ∞. This means that the DTFT and DFT describe frequency components in the signal averaged over all time. Interesting signals like music and speech are characterized the ways in which frequency components change over time. (These components could represent objects such as the phonemes that constitute a spoken word or the notes that constitute a musical composition.) This observation motivated the development of the *short-time Fourier transform* (STFT).

The STFT considers only a short-duration segment of a longer signal and computes its Fourier transform. Typically this is accomplished by multiplying a longer time function $x[n]$ by a window function $w[n]$ that is brief in duration. Two commonly-used finite-duration windows are the rectangular window, which essentially extracts only the desired short sequence without further modification, and the Hamming window, which applies a taper to the ends to improve the representation in the frequency domain. If a continuous frequency variable ω is used (as in the DTFT), the STFT can be described as

$$X[n,\omega) = \sum_{m=-\infty}^{\infty} w[n-m]x[m]e^{-j\omega m}$$

In principle, the window could be either finite or infinite in duration, and in the latter case exponential windows are popular.

If discrete frequency variables $\omega_k = 2\pi k/N$ and a finite-duration window with nonzero values of n from 0 to $N-1$, the equation becomes

$$X[n,k] = \sum_{m=n-(N-1)}^{n} w[n-m]x[m]e^{-j\omega_k m} = \sum_{m=n-(N-1)}^{n} w[n-m]x[m]e^{-j2\pi mk/N}$$

Note that $X[n,k]$ is a function of both time and frequency. The variable n denotes the location of the analysis window along the time axis, and the segment of time delimited by the window is frequently referred to as the *analysis frame*. The variable k is a frequency index, and is sometimes

referred to as a *frequency bin*. We can think of the STFT as representing the DFT of the finite-duration time function $x[m]w[n-m]$. Here the variable m is a "dummy time argument and the variable n identifies the location of the short segment of the original time function as it is extracted using the window $w[-m]$, which moves along the m-axis according to the value of n.

A. Impact of window size and shape

We can formalize the interaction between the original time function, the window size and shape, and the resulting STFT as follows. Using the continuous-frequency version of the STFT, we obtain

$$w[n-m] \Leftrightarrow \sum m = -\infty^{\infty} w[n-m]e^{-j\omega m}$$

Letting $l = n - m$ and $m = n - l$ we obtain

$$w[n-m] \Leftrightarrow \sum_{m=-\infty}^{\infty} w[l]e^{-j\omega(n-l)} = e^{-j\omega m}W(e^{-j\omega})$$

Hence,

$$w[n-m]x[m] \Leftrightarrow \frac{1}{2\pi}(W(e^{-j\omega})e^{-j\omega n)} \circledast W(e^{j\omega})$$

In other words, the STFT can also be thought of as the circular convolution in frequency of the Fourier transform of the original input signal with the Fourier transform of the window function. Consequently, a brief analysis window $w[n]$ will give us good temporal resolution at the expense of a lot of blurring in frequency, while a broader temporal window will provide sharp spectral resolution at the expense of temporal resolution. This applies as well to the discrete-frequency implementation of the STFT as well.

B. Inversion of the STFT

As we have stated, we can think of the STFT as the Fourier transform of the windowed time function:

$x[m]w[n-m] \Leftrightarrow X[n,\omega)$ so

$$w[n-m]x[m] = \frac{1}{2\pi}\int_{-\infty}^{\infty} X[n,\omega)e^{j\omega n}d\omega$$

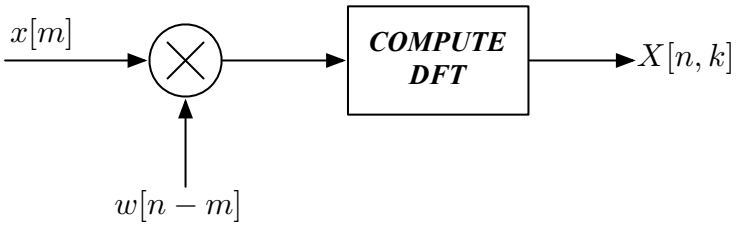
For $n = m$ we can write

$$x[n]w[0] = \frac{1}{2\pi}\int_{-\infty}^{\infty} X[n,\omega)e^{j\omega n)}d\omega$$

Hence the only absolute constraint for being able to recover $x[n]$ from $X[n,\omega)$ is that $w[0] \neq 0$.

II. Alternate interpretations of the STFT operation

A. Fourier transform implementation



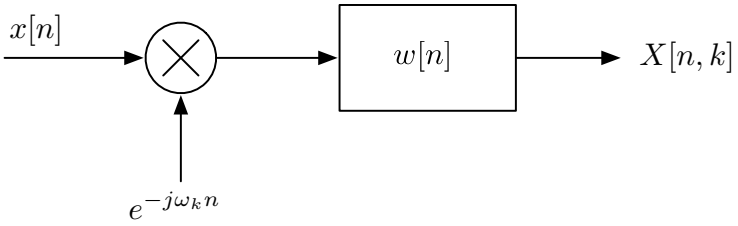
As discussed in the section above, the most straightforward way of thinking about the calculation of the STFT is as a multiplication of the time function $x[m]$ by a finite-duration window function $w[m]$ and computation of the Fourier transform of their product:

$$X[n,k] = \sum_{m=n-(n-1)}^{n} (x[m]w[n-m])e^{-j2\pi mk/N} = \sum_{m=n-(n-1)}^{n} (x[m]w[n-m])e^{-j\omega_k m}$$

This is referred to as the *Fourier transform implementation* of the STFT.

As stated above, the time function is multiplied by a window that shifts along the m-axis, and that the variable n indicates the position of the window, which designates the location of the "analysis frame. We can think of this as multiplying the time function $x[m]$ by a short-time window $w[n-m]$ that is located at the sample n.

B. Lowpass filter interpretation of the STFT



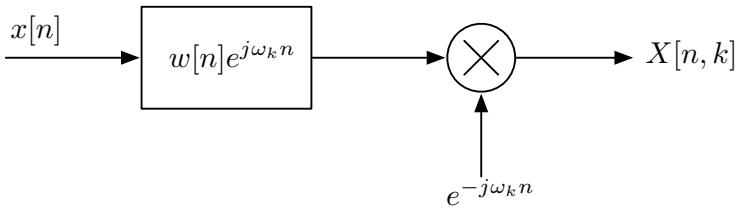
We can rearrange the terms of the STFT equation slightly to produce

$$X[n,k] = \sum_{m=n-(N-1)}^{n} w[n-m](x[m]e^{-j\omega_k m})$$

This corresponds to multiplying the signal $x[n]$ by the complex exponential function $e^{-j2\pi nk/N}$ and passing the product through a filter with unit sample response $w[n]$. This implementation of the STFT, which is referred to as the *lowpass filter implementation*, is depicted in the figure above.

The unit sample responses of all typical windows are *lowpass* in nature, as discussed in class. Note that multiplying $x[n]$ by $e^{-j\omega_k n}$ shifts the spectrum to the *left* by ω_k, so that the components that were originally at frequency ω_k now lie at frequency 0. Hence, if we consider a particular value of k, the STFT $X[n,k]$ reflects the smoothed frequency content of the original function $x[n]$ at ω_k as it evolves over time (represented by the variable n). As noted above, the lowpass filter implementation is mathematically equivalent to the Fourier transform implementation of the STFT $X[n,k]$.

C. Bandpass filter interpretation of the STFT



A third mathematically-equivalent interpretation can be obtained by a simple algebraic change of variables which yields another form of the STFT equation:

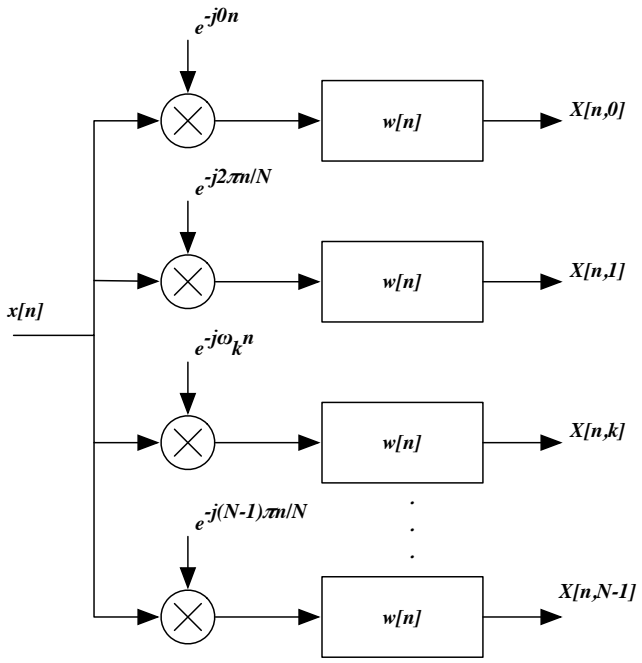
$$X[n,k] = e^{-j\omega_k m} \sum_{m=n-(N-1)}^{n} \left(w[n-m]e^{-j\omega_k (n-m)} \right) x[m]$$

This implies that the input signal is passed through a bandpass filter consisting of the original lowpass window filter, frequency shifted so that it now passes frequency components centered around ω_k . The spectrum of the output is then translated to the left so that the components of $X[n,k]$ that were originally at frequency ω_k are ultimately centered around frequency zero. This interpretation is shown in the figure above and is referred to as the *bandpass filter implementation.*

Keep in mind that all three interpretations of the STFT are mathematically equivalent and that they all characterize $X[n,k]$ as representing the smoothed frequency content of the original function $x[n]$ at ω_k, evolving over time.

III. Implementations of the STFT using real time functions and impulse responses

The original lowpass and bandpass filter implementations assume multiplication by complex exponentials, and (in the bandpass filter implementation) filters with complex unit sample responses. For example, the lowpass filter implementation described in Sec. II. B above can be illustrated as



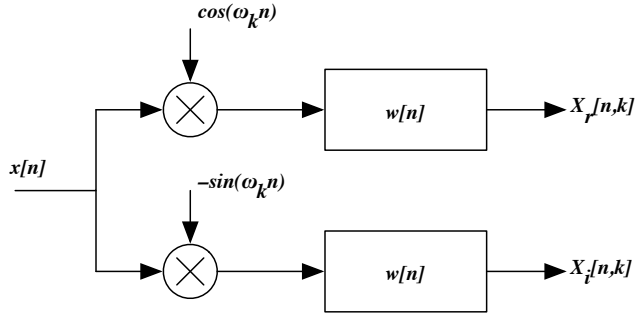
Each channel in the above diagram can be written as

$$X[n,k] = w[n] * \left(x[n]e^{-j\omega_k n}\right) = w[n] * (x[n]\cos(\omega_k n)) - jw[n] * (x[n]\sin(\omega_k n))$$

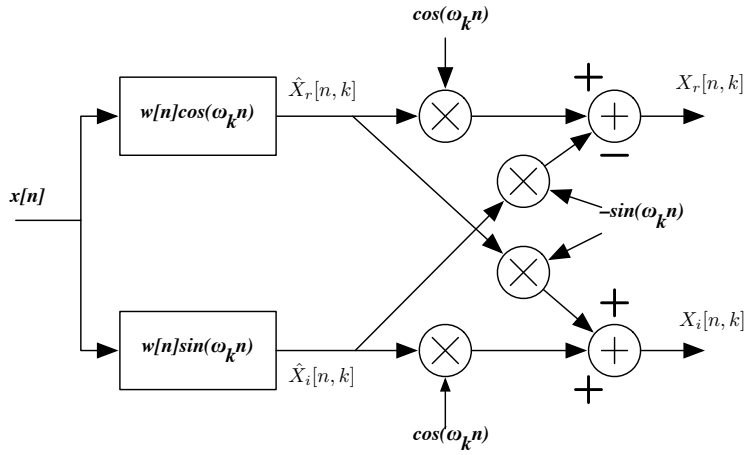
which we can rewrite as

$$X[n,k] = X_r[n,k] + jX_i[n,k]$$

Note that the functions $X_r[n,k]$ and $X_i[n,k]$ are both real. This can be illustrated as



The corresponding bandpass filter implementation is a bit more algebraically involved, but can be obtained using similar principles. It can be illustrated as in the diagram below. (Note that Figs. 6.15 and 6.16 in Lim and Oppenheim use different sign conventions.)

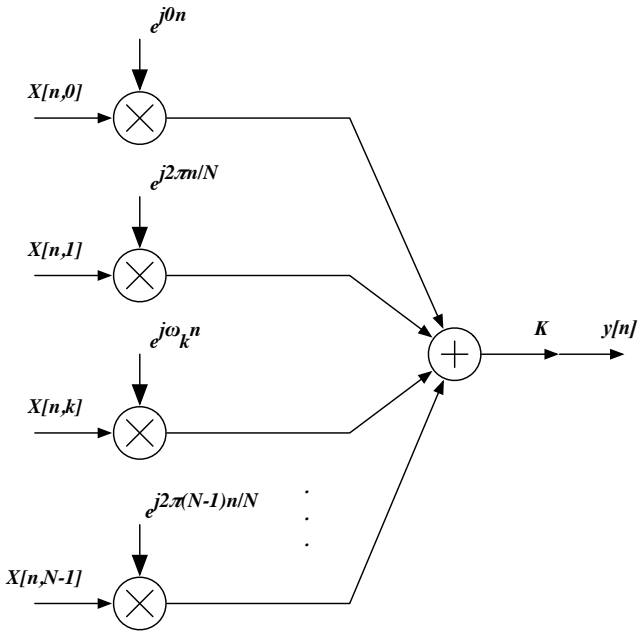


Note that the functions $\hat{X}_r[n,k]$ and $\hat{X}_i[n,k]$ (immediately after the cosine multiplys) have magnitudes that are equal to the final outputs $X_r[n,k]$ and $X_i[n,k]$. If we are only interested in computing the short-time *magnitude* spectrum (which is frequently the case), the bandpass implementation is more computationally efficient than the lowpass implementation because the multiplys are folded into the filtering.

IV. Short-time Fourier synthesis

We will consider two methods of recovering the time function $x[n]$ from the STFT $X[n,k]$, one based on the filtering implementation and the other based on the Fourier transform implementation. We will first consider the filterbank implementation.

A. The Filter Bank Summation (FBS) method



Let us assume that $w[n]$ is of finite duration, and that $w[n] \neq 0$ for$0 \leq n \leq N-1$.

Because

$$\sum_{m=n-(n-1)}^{n} (x[m]w[n-m])e^{-j\omega_k m} \text{ where as usual } \omega_k = 2\pi k/N$$

we can write

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

Hence, for $w[0] \neq 0$ we can write

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

This implies that in principle we can obtain the time function by multiplying the various short-time Fourier transform coefficients $X[n,k]$ by the function $e^{j\omega_k n}$, adding all the products together, and dividing by $Nw[0]$.

Are there any constraints on the window size and shape that are required for this to work? Let us designate the output of a complete analysis-synthesis system as $y[n]$. Then we can write

$$y[n] = \frac{1}{Nw[0]}\sum_{k=0}^{N-1} X[n,k]e^{j\omega_k n} = \frac{1}{Nw[0]}\sum_{k=0}^{N-1}\left(\sum_{m=-\infty}^{\infty} x[m]w[n-m]e^{-j\omega_k m}\right)e^{j\omega_k n}$$

Since the argument of the inner sum can be written as $x[m]w[n-m]e^{j\omega_k(n-m)m}$, the expression can be rewritten as

$$y[n] = \frac{1}{Nw[0]} \sum_{k=0}^{N-1} x[n] * \left(w[n]e^{j\omega_k m}\right) = \frac{1}{Nw[0]} x[n] * \left(w[n] \sum_{k=0}^{N-1} e^{j\omega_k m}\right)$$

In order for $y[n]$ to be at least proportional to $x[n]$, we would need for the expression inside the parentheses to be proportional to $\delta[n$. As you know, the sum can be written as

$$\sum_{k=0}^{N-1} e^{j2\pi nk/N} = \frac{1 - e^{j2\pi N}}{1 - e^{j2\pi n/N}}$$

which is equal to 1 for $n = rN$ for integer r and zero otherwise. Hence we must require that $w[n] = 0$ for $n = rN$ in order for the filter-bank summation (FBS) method of synthesis to be used. This condition is sometimes called the "FBS constraint. The FBS constraint is satisfied trivially for finite-duration windows of length N, but it is also satisfied by another class of windows known as *Nyquist windows* such as the familiar $\sin(x)/x$ and $\sin(Nx)\sin(x)$ functions.

Another way of expressing the FBS constraint is that we require that $w[n]$ satisfy the constraint

$$w[n]p_N[n] = \delta[n]$$

where N is the number of equally-spaced frequency channels and $p_N[n]$ is a pulse train with period N:

$$p_N[n] = \begin{cases} 1, & n = rN \\ 0, & \text{otherwise} \end{cases}$$

This, of course is simply a compact way of stating that $w[n] = 0$ for $n = rN$ and $n = 0$ otherwise. Taking the DTFT of the constraint equation above produces

$$\frac{1}{2\pi} W(e^{j\omega}) \circledast P_N(e^{j\omega}) = 1$$

where $P_N(e^{j\omega})$ is the DTFT of $p_N[n]$ and the symbol $\circledast$ indicates circular convolution as before.

As you will recall, the DTFT $P_N(e^{j\omega})$ is an infinite train of delta functions in frequency, each with area $2\pi/N$, separated by $2\pi/N$ along the frequency axis. This means that an alternate form of the FBS constraint is

$$\sum_{l=0}^{N-1} W(e^{j(\omega - 2\pi l/N)}) = 1$$

In other words, the FBS constraint is satisfied if the DTFTs of the window translated every $2\pi/N$ radians along the ω-axis and added together sum to 1.

B. The Overlap-Add (OLA) method

The overlap-add (OLA) is easier to describe and analyze than the FBS method. In brief, it is as follows:

1. Sample $X[n,k]$ at intervals $n = rR$, with r and R integers
2. Compute the inverse DTFTs of $X[rR,\omega) \equiv Y_r(e^{j\omega})$
3. Add these IDTFTs together, staggering them at intervals of R to produce the final output $y[n]$

In other words, we write

$$y[m] = \sum_{r=-\infty}^{\infty} \frac{1}{M} \sum_{k=0}^{N-1} Y_r(e^{j\omega_k}) e^{j\omega_k n} = \sum_{r=-\infty}^{\infty} x[m]w[rR-m] = x[m] \sum_{r=-\infty}^{\infty} w[rR-m]$$

For $y[n]$ to be proportional to $x[n]$ we must ensure that the sum of all the window functions

$$\sum_{r=-\infty}^{\infty} w[rR-m]$$

is a constant. This is satisfied, for example, by rectangular windows that are abutted and Hamming windows that are overlapped by 50 percent.

V. Sampling in time and frequency

As in many other engineering applications, we are interested in being as efficient in computation and stor- age as possible. We briefly discuss in this section the number of numbers that are (nominally) needed to obtain a complete characterization of a time function using short-time Fourier analysis. In this section we briefly discuss the issues involved with sampling in time and frequency, specifically reviewing how many frequency channels we need and how many sparse we can sample in time for a given window $w[n]$. As will see, the answer to these questions can depend on whether OLA or FBS synthesis is used. We will consider only FIR windows at this time, although the same principles hold for IIR windows as well.

A. The Fourier-transform implementation with overlap-add synthesis

Let us consider first the Fourier-transform implementation with overlap-add synthesis. As has been discussed above, time sampling using OLA is determined by the length and shape of the window. Specifically, the spacing must be such that the sum of the windows (in the locations that they occur at) must add to a constant. This means, for example, that if we have a rectangular window with length N_w, successive windows can be abutted, causing the window spacing to be N_w. If Hamming windows are used, on the other hand, a 50% overlap will be needed, causing the windows to be spaced apart by $N_w/2$ samples. Since we are computing DFTs for each window, we need a DFT

size that is at least as large as the duration of the window, or at least N_w channels. Hence, for a sampling rate of F_s and a Hamming window of length N_w the total number of samples per second would be the sampling rate times the number of channels divided by the spacing of the windows. For Hamming windows this would be

$$N_s = \frac{F_s N_w}{N_w/2} = 2F_s$$

For either filterbank implementation with FBS resynthesis, the number of channels is determined by the FBS constraint that for a given window length N_w the window shape $w[n]$ must be such that

$$w[n] = 0 \text{ for } n = rN_w \text{ with } r \neq 0$$

This constraint is automatically satisfied for an FIR window if the number of channels N is greater than or equal to N_w. The time sampling is determined by the effective bandwidth of the window. This is determined by looking for the first frequency ω at which the Fourier transform of the window, $W(e^{j\omega})$ is zero. For a rectangular window this frequency is $2\pi/N_w$ and for the Hamming window this frequency is $4\pi/(N_w - 1)$. Recall from our discussion of multi-rate DSP that if a discrete-time signal is limited to frequencies of π/M_d, we can downsample it by a factor of M_d. Hence, for a sampling rate of F_s and a downsampling rate of M_d, the total number of samples per second would be the sampling rate times the number of channels divided by the decimation rate, which for Hamming windows would be

$$N_s = \frac{F_s N_w}{(N_w - 1)/4} \approx 4F_s$$

As can be seen, the number of numbers per second using FBS resynthesis is twice as many as with OLA resynthesis, which in turn is twice as many as the original sampled waveform in the time domain. Nevertheless, the STFA representation is widely used because of the insight it can provide in analyzing signals as well as the signal-manipulation operations that it enables.