3. Short-Time Fourier Transforms

Traditional Fourier transforms describe the frequency components in a signal averaged over all time. The most important and informative aspects of signals like speech and music, however, is how these frequency components evolve over time. In this chapter we discuss short-time Fourier transforms, which enable us to characterize signals with time-varying frequency components.

3.1 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 individual notes that constitute a musical composition.) These observations motivated the development of the *short-time Fourier transform* (STFT).[1]

3.2 Computing the short-time Fourier transform

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 the continu-

[1] Most of these discussions follow the treatment of the material in the books on speech signal processing by Rabiner and Schafer (1978, 2010).

ous frequency variable ω is used (as in the DTFT), the STFT can be described as

$$X[n,\omega] = \sum_{m=-\infty}^{\infty} w[n-m]x[m]e^{-j\omega m} \tag{3.1}$$

In principle, the window could be either finite or infinite in duration, and in the latter case exponential windows are popular. The delimiters in $X[n,\omega)$ are unconventional, of course, and this usage is intended to highlight the fact that the frame index n is discrete while the frequency variable ω is continuous.

In practice, it is common to evaluate the STFT at only a finite set of equally-spaced points along the frequency axis, just as the DFT is frequently used instead of the DTFT in conventional applications of digital signal processing. We will use the variable N to specify the number of discrete frequency channels used in the STFT, and the variable N_w to specify the length of the window function $w[n]$ when it is finite in duration. Using these notational conventions, the frequencies over which the STFT is evaluated become $\omega_k = 2\pi k/N$. With a finite-duration window with nonzero values of n from 0 to $N_w - 1$, the STFT equation becomes

$$X[n,k] = \sum_{m=n-(N_w-1)}^{n} w[n-m]x[m]e^{-j\omega_k m} = \sum_{m=n-(N_w-1)}^{n} w[n-m]x[m]e^{-j2\pi mk/N} \tag{3.2}$$

Note that $X[n,k]$ is a function of both time and frequency and now both the time and frequency variables are discrete. 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 $w[n-m]$, which moves along the m-axis according to the value of n.

3.2.1 Impact of window size and shape

Let us begin by turning our attention to the impact of the window shape and duration on the nature of the STFT. We can formalize the interaction between the original time function, the window size and shape, and the resulting STFT as follows. Considering first the continuous-frequency version of the STFT, recall that $X[n,\omega)$ is the DTFT of the input function $x[n]$ multiplied by the window function. The Fourier transform of $w[n-m]$ is

$$w[n-m] \Leftrightarrow \sum_{m=-\infty}^{\infty} w[n-m]e^{-j\omega m} \tag{3.3}$$

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

$$w[n-m] \Leftrightarrow \sum_{l=-\infty}^{\infty} w[l]e^{-j\omega(n-l)} = e^{-j\omega n}W(e^{-j\omega}) \tag{3.4}$$

Hence,

$$w[n-m]x[m] \Leftrightarrow \frac{1}{2\pi}(W(e^{-j\omega})e^{-j\omega n}) \circledast X(e^{j\omega}) \tag{3.5}$$

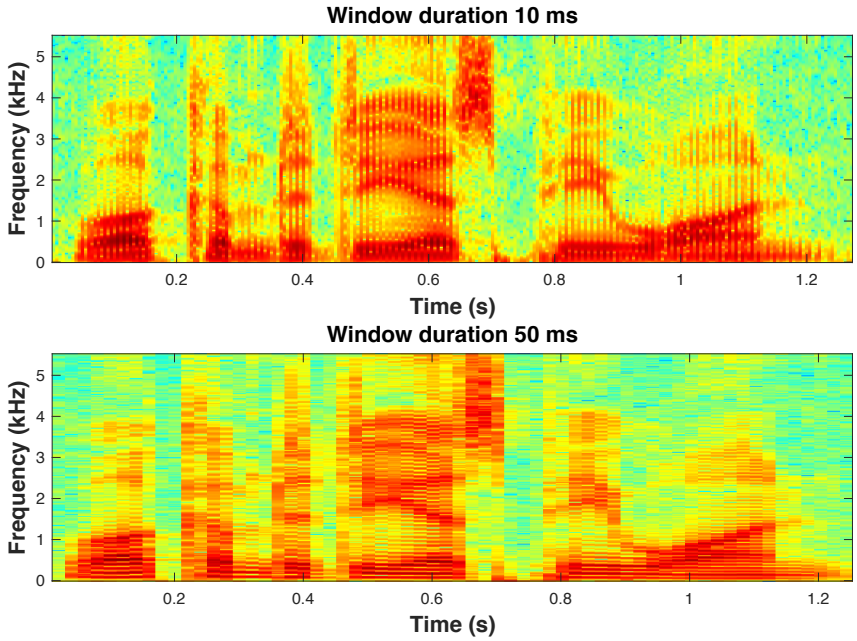

Figure 3.1: Impact of window duration on the STFT. A spectrogram of a brief utterance is shown, using Hamming windows of duration 10 ms (upper panel) and 50 ms (lower panel).

where the symbol $\circledast$ indicates circular convolution.

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 time reversed and shifted. Because briefer functions in time produce broader Fourier transforms in frequency, the use of 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 reduced temporal resolution. This applies to the discrete-frequency implementation of the STFT as well.

The tradeoff between temporal and spectral resolution is illustrated in Fig. 3.1. Figure 3.1 shows two examples of a *spectrogram* of a brief utterance ("Welcome to DSP-I") spoken by the author. The horizontal axis represents time while the vertical axis represents frequency. For now it is sufficient to note that the speech waveform can be modeled by a filtered pulse train called *glottal pulses*, which are generated by the vocal chords with time-varying fundamental frequency. The glottal pulses are input to an acoustic filter with time-varying frequency response that is shaped by the configuration of the throat, tongue, and lips, etc. (It should also be noted that some phonemes such as /s/ , /f/, and /th/ are produced by exciting the acoustic filter with broadband noise instead of the quasi-periodic glottal pulses.) As the colors in the display go from blue to green to yellow to orange to red for a particular spectro-temporal element, the power of the signal becomes greater for that frequency and time. In fact, the desire to analyze, display, interpret, and manipulate the time-varying characteristics of speech and music in a useful fashion has been the prime motivation toward the development of the mathematics that

are the basis for the STFT.

In the example of Fig. 3.1, we can observe that the /c/ sound in "welcome" occurs just after 0.2 seconds and the /s/ in "DSP" occurs at about 0.65 seconds. The window functions used in the figure are Hamming windows of duration 10 ms (upper panel) and 40 ms (lower panel). The fundamental frequency of the vocal tract pulses varies, but it is about 100 Hz, and the corresponding period is about 10 ms. The windows are overlapped by 50 percent for reasons to be discussed below. The image in the upper panel appears to show vertical bars, which occur because the window duration is comparable to the 10-ms period of the glottal pulses, so some of the so-called *analysis frames* occur at the time of the glottal pulses and some of them occur between them. In contrast, horizontal bars are seen in the lower panel of Fig. 3.1, which is computed using windows of duration 50 ms. In this case, the window duration is long enough so that the window smears over successive glottal pulses, but the frequency resolution is now sufficiently fine that the horizontal bars appear at analysis frequencies that are multiples of the fundamental frequency, which is about 100 Hz. It can be seen that the separation of the vertical bars in the upper panel of Fig. 3.1 is approximately .01 seconds and the separation of the horizontal bars in the lower panel is approximately 100 Hz.

In practice, the duration of the analysis window is set according to the needs of the application. For example, the window duration is typically between 20 and 35 ms for automatic speech recognition, but longer than that (75-120 ms) for speaker identification.

3.2.2 Inversion of the STFT

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

$$x[m]w[n-m] \Leftrightarrow X[n,\omega) \text{ so} \tag{3.6}$$

$$w[n-m]x[m] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X[n,\omega)e^{j\omega m}d\omega \tag{3.7}$$

For $n = m$ we can write

$$x[n]w[0] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X[n,\omega)e^{j\omega n}d\omega \tag{3.8}$$

or, solving for $x[n]$,

$$x[n] = \frac{1}{2\pi w[0]}\int_{-\pi}^{\pi} X[n,\omega)e^{j\omega n}d\omega \tag{3.9}$$

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

3.3 Alternate interpretations of the STFT operation

Although so far we have talked about the STFT simply as being the DTFT or DFT of a time function after it is multiplied by a sliding window in time, there are two other mathematically-equivalent ways of formulating the STFT. We discuss and compare the three implementations of the STFT in this section. We will use the DFT-based formulation of the STFT in this section because it is this formulation that is most commonly used in practice.

3.3.1 Fourier transform interpretation of the STFT

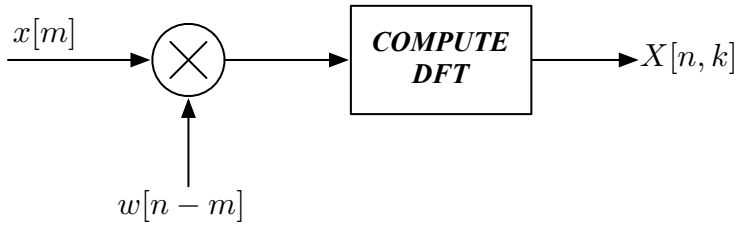

Figure 3.2: The Fourier transform implementation of the STFT.

As discussed 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- or infinite-duration window function $w[n-m]$, followed by the computation of the Fourier transform of their product:

$$X[n,k] = \sum_{m=n-(N_w-1)}^{n} (x[m]w[n-m])e^{-j2\pi mk/N} = \sum_{m=n-(N_w-1)}^{n} (x[m]w[n-m])e^{-j\omega_k m} \qquad (3.10)$$

In the expression above, which assumes a finite-duration window of length N_w, the variable n indicates the position of the window, which designates the location of the "analysis frame." The variable k refers to the *frequency bin* in question. This is referred to as the *Fourier transform implementation* of the STFT.

3.3.2 Lowpass filter interpretation of the STFT

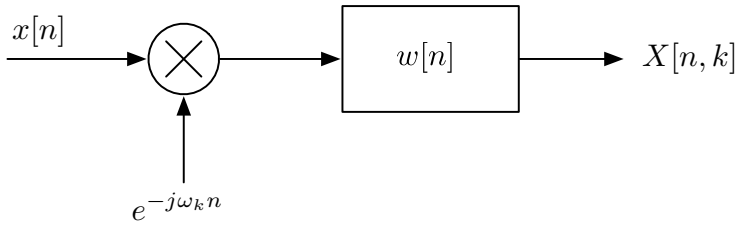

Figure 3.3: The lowpass filter implementation of the STFT.

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

$$X[n,k] = \sum_{m=n-(N_w-1)}^{n} w[n-m]x[m]e^{-j\omega_k m} = \sum_{m=n-(N_w-1)}^{n} w[n-m](x[m]e^{-j\omega_k m}) \qquad (3.11)$$

This corresponds to multiplying the signal $x[n]$ by the complex exponential function $e^{-j2\pi nk/N} = e^{-j\omega_k n}$ 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 Fig 3.3 above.

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. Because the unit sample responses of typical windows are lowpass in nature, the STFT $X[n,k]$ reflects the smoothed frequency content of the original function $x[n]$ at frequency 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 $X[n,k]$.

3.3.3 Bandpass filter interpretation of the STFT

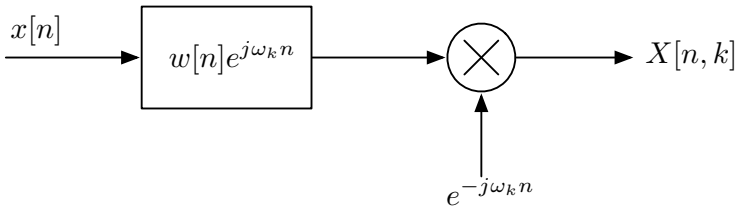

Figure 3.4: The bandpass filter implementation of the STFT.

A third mathematically-equivalent interpretation can be obtained by a simple manipulation of the STFT expression:

$$X[n,k] = \sum_{m=n-(N_w-1)}^{n} w[n-m]x[m]e^{-j\omega_k m} = \left(\sum_{m=n-(N_w-1)}^{n} \left(w[n-m]e^{j\omega_k (n-m)} \right) x[m] \right) e^{-j\omega_k n} \tag{3.12}$$

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 frequency 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 Fig. 3.4 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.

3.3.4 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 the filters have complex unit sample responses. For example, the lowpass filter implementation described in Eq. (3.11) above can be illustrated as

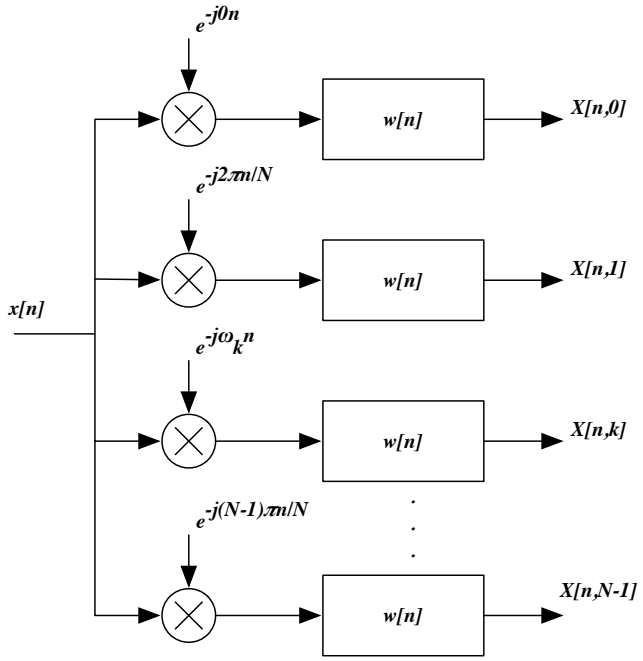

Figure 3.5: The lowpass filter STFT implementation on a channel-by-channel basis.

Each channel in Fig. 3.5 can be written as

$$X[n,k] = w[n] * \left(x[n]e^{-j\omega_k n}\right) = w[n] * (x[n]\cos(\omega_k n)) - jw[n] * (x[n]\sin(\omega_k n)) \tag{3.13}$$

which we can rewrite as

$$X[n,k] = X_r[n,k] - jX_i[n,k] \tag{3.14}$$

Note that the functions $X_r[n,k]$ and $X_i[n,k]$ are both real. This system is illustrated in Fig. 3.6

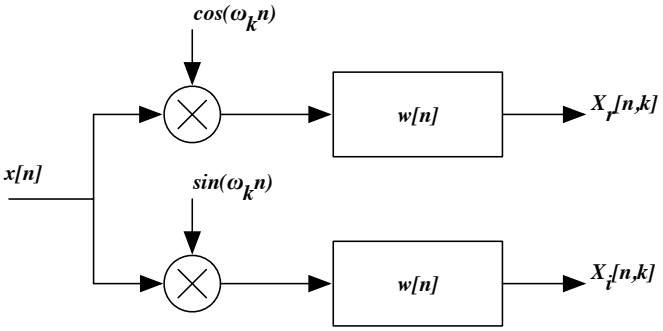

Figure 3.6: Single channel of the lowpass STFT implementation using real sample responses.

The corresponding bandpass filter implementation is a bit more algebraically involved, but can be obtained using similar principles, as is seen in Fig. 3.7.

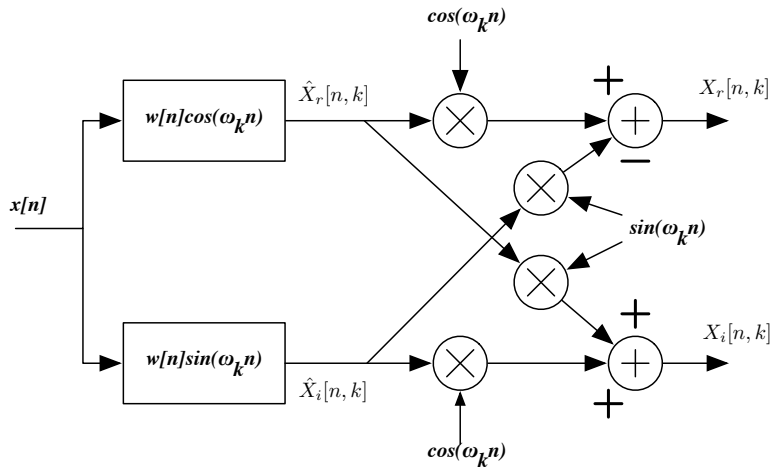

Figure 3.7: Single channel of the bandpass STFT implementation using real sample responses.

Note that the functions $\hat{X}_r[n,k]$ and $\hat{X}_i[n,k]$ (immediately before the cosine multiplications) 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), we can eliminate all the computation after the filter outputs, which causes the bandpass implementation to be more computationally efficient than the lowpass implementation because the multiplications are folded into the filtering.

3.4 Downsampling the STFT

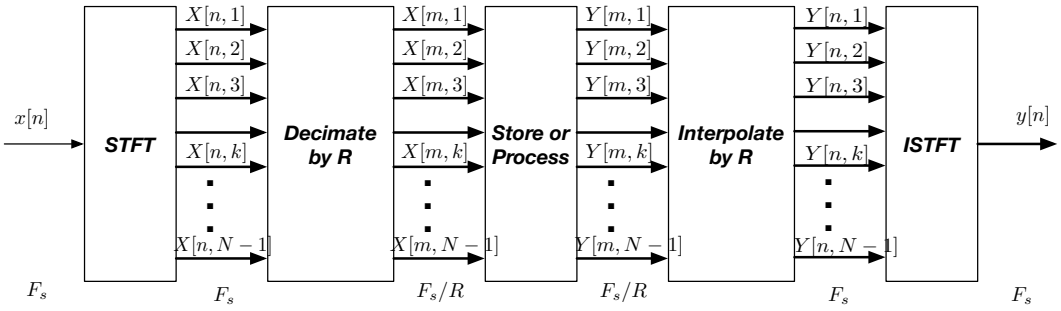

Figure 3.8: Block diagram of a complete STFT analysis/synthesis with downsampling.

In practice, the STFT coefficients are frequently downsampled as depicted in Fig. 3.8, either for efficiency in computation or for efficiency in manipulation or modification of the representation in the STFT domain. This is possible because STFT coefficients in a fixed frequency bin evolve slowly as a function of time. It is easiest to understand how this happens by reviewing the lowpass filter STFT implementation, as depicted in Fig. 3.3, which describes the STFT calculation as shifting the spectrum to the left by ω_k radians and then lowpass filtering the frequency-shifted signal by a filter having the unit sample response of $w[n]$. As an example, let us consider the Hamming window, which

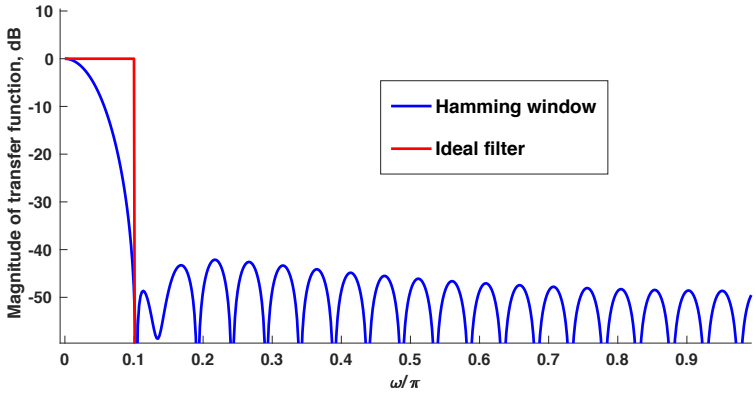

Figure 3.9: Magnitude in dB of the DTFT of a Hamming window of length 41 samples (blue curve). The red curve is the response of the corresponding ideal filter.

is frequently (although far from exclusively) used as the window function for STFTs. It is well known that the main lobe of the DTFT of the Hamming window has a mainlobe width of $8\pi/(N_w - 1)$ radians (including both positive and negative frequencies) where N_w is the length of the window. For example, a Hamming window of length 41 would have its first zero crossing in positive frequency at $\pi/10$ radians, which suggests that the output of this filter could be downsampled by a factor of 10.

Figure 3.9 compares the actual magnitude of the transfer function of a Hamming window of length 41 (blue curve) with the corresponding ideal filter (red curve). It is clear that the response of the Hamming window itself used as a lowpass filter is far from ideal, either in terms of the flatness of the response in the "passband" or in the suppression of sidelobes in the "stopband." Nevertheless, this approximation is commonly used in short-time Fourier transforms and normally it works well enough. And, in general, the location of the first zero-crossing in the frequency response of the window function is typically used as the nominal cutoff frequency for windows of other shapes as well.

3.5 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 implementations and the other based on the Fourier transform implementation. We will first consider the filterbank implementation.

3.5.1 The Filter Bank Summation (FBS) method

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

$$X[n,k] = \sum_{m=n-(N_w-1)}^{n} (x[m]w[n-m])e^{-j\omega_k m} \tag{3.15}$$

where as usual $\omega_k = 2\pi k/N$, we can write

$$w[n-m]x[m] = \frac{1}{N}\sum_{k=0}^{N-1} X[n,k]e^{j\omega_k m} \tag{3.16}$$

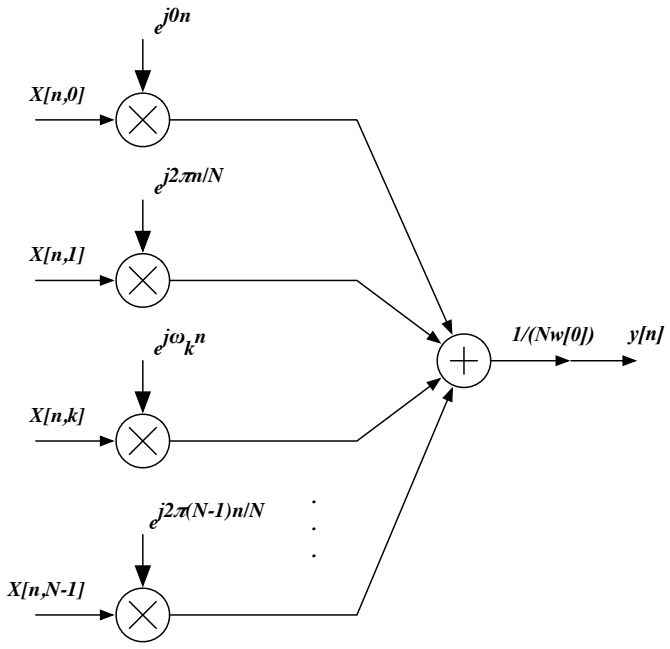

Figure 3.10: Block diagram of the filterbank summation (FBS) method of STFT synthesis.

Hence, for $w[0] \neq 0$ we obtain

$$x[n] = \frac{1}{Nw[0]} \sum_{k=0}^{N-1} X[n,k] e^{j\omega_k n} \tag{3.17}$$

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

$$y[n] = \frac{1}{Nw[0]} \sum_{k=0}^{N-1} X[n,k] e^{j\omega_k n} = \frac{1}{Nw[0]} \sum_{k=0}^{N-1} \left(\sum_{m=-\infty}^{\infty} x[m] w[n-m] e^{-j\omega_k m} \right) e^{j\omega_k n} \tag{3.18}$$

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

$$y[n] = \frac{1}{Nw[0]} \sum_{k=0}^{N-1} x[n] * \left(w[n] e^{j\omega_k n} \right) = \frac{1}{Nw[0]} x[n] * \left(w[n] \sum_{k=0}^{N-1} e^{j\omega_k n} \right) \tag{3.19}$$

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, we can write the inner sum as

$$\sum_{k=0}^{N-1} e^{j2\pi nk/N} = \frac{1 - e^{j2\pi n}}{1 - e^{j2\pi n/N}} \tag{3.20}$$

which is equal to N for $n = rN$ for integer r and zero otherwise. Hence we must require that $w[n] = 0$ for $n = rN$ and $n \neq 0$ 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, which have an infinite number of equally-spaced repeating zeros in the time function.

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

$$w[n]p_N[n] = \delta[n] \tag{3.21}$$

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

$$p_N[n] = \begin{cases} 1, & n = rN \\ 0, & \text{otherwise} \end{cases} \tag{3.22}$$

This, of course is simply a compact way of stating that $w[n] = 0$ for $n = rN$ and $n = 0$ otherwise, as before. Taking the DTFT of Eq. (3.21) above produces

$$\frac{1}{2\pi} W(e^{j\omega}) \circledast P_N(e^{j\omega}) = 1 \tag{3.23}$$

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

$$\frac{1}{N} \sum_{l=0}^{N-1} W(e^{j(\omega - 2\pi l/N)}) = \text{some constant} \tag{3.24}$$

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 a constant that is independent of ω.

3.5.2 The Overlap-Add (OLA) method

The overlap-add (OLA) method, which is based on the Fourier transform implementation of the STFT, is easier to describe and analyze than the FBS method. In our discussion here we will make use of the DTFT-based definition of the STFT, but the OLA method using the DFT-based definition works in exactly the same way.

Consider samples of the STFT spaced apart by R frames, $X[rR, \omega)$, which we will also refer to as $Y_r(e^{j\omega})$. The inverse DTFT of $Y_r(e^{j\omega})$ is $w[rR - m]x[m] \equiv y_r[m]$, as defined in Eq. (3.7). We obtain the output function $y[m]$ by simply adding all of the inverse transforms $y_r[m]$ together:

$$y[m] = \sum_{r=-\infty}^{\infty} y_r[m] = \sum_{r=-\infty}^{\infty} w[rR - m]x[m] = x[m] \sum_{r=-\infty}^{\infty} w[rR - m] \tag{3.25}$$

Clearly, for $y[n]$ to be equal to $x[n]$ we must ensure that the sum of all the window functions equals 1:

$$\sum_{r=-\infty}^{\infty} w[rR-m] = 1 \tag{3.26}$$

This is satisfied, for example, by rectangular windows that are abutted and triangular or Hamming windows that are overlapped by 50 percent, as depicted in Fig. 3.11, among many other window shapes and spacings.

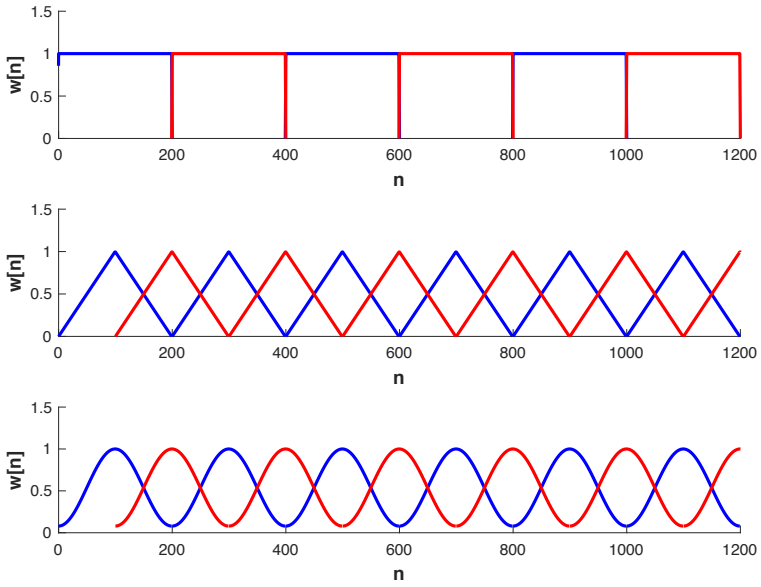

Figure 3.11: Examples of sequences of rectangular, triangular (Bartlett), and Hamming window functions of length 200 that sum to 1. Note that the the rectangular windows may be abutted, while the triangular and Hamming windows must be overlapped by 50 percent.

3.6 Sampling in time and frequency

As in many other engineering applications, we are interested in being as efficient in computation and storage 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, specifically reviewing how many frequency channels we need and how sparsely 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.

3.6.1 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, as discussed above. 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, requiring 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 to avoid temporal aliasing, or at least N_w channels. Now let us define the constant N_s to represent the total number of numbers per second of input needed to store the signal using the STFT representation. For a sampling rate of F_s and a rectangular window of length N_w the N_s would be equal to the sampling rate times the number of channels divided by the spacing of the windows. For rectangular windows this would be

$$N_s = \frac{F_s N_w}{N_w} = F_s \tag{3.27}$$

For Hamming windows this would be

$$N_s = \frac{F_s N_w}{(N_w/2)} = 2F_s \tag{3.28}$$

3.6.2 The lowpass and bandpass implementations with filterbank-summation synthesis

For either the lowpass or bandpass 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

$$w[n] = 0 \text{ for } n = rN_w \text{ with } r \neq 0 \tag{3.29}$$

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, as shown in Fig. 3.9. 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 rectangular windows would be

$$N_s = \frac{F_s N_w}{(N_w/2)} = 2F_s \tag{3.30}$$

For Hamming windows the total number of samples per second would be

$$N_s = \frac{F_s N_w}{(N_w - 1)/4} \approx 4F_s \tag{3.31}$$

As can be seen, the number of numbers per second using FBS resynthesis needed to represent a signal is twice as many as with OLA resynthesis, which in turn is twice as many as the original sampled waveform in the time domain. In fact, the representation is even

more inefficient because the STFT coefficients are complex, requiring two real numbers each, although if the time function is real, the coefficients representing positive and negative frequencies would be complex conjugates of each other. Nevertheless, the STFT representation is widely used in all cases because of the insight it can provide in analyzing signals as well as the signal-manipulation operations that it enables.

3.7 Applications of short-time Fourier analysis

3.7.1 Phase vocoding

Phase vocoding was originally developed by Flanagan and Golden (1966) as a technique to accomplish high-quality speech coding. While phase vocoding did not prove to be a commercially-successful method to encode speech, it does have a number of properties that make it useful for expanding and contracting speech in time, and for changing the pitch of music with relatively small changes in the musical timbre. In this section we describe the basic principles of phase vocoding and describe how it is applied to nonlinear transformations in time and frequency.

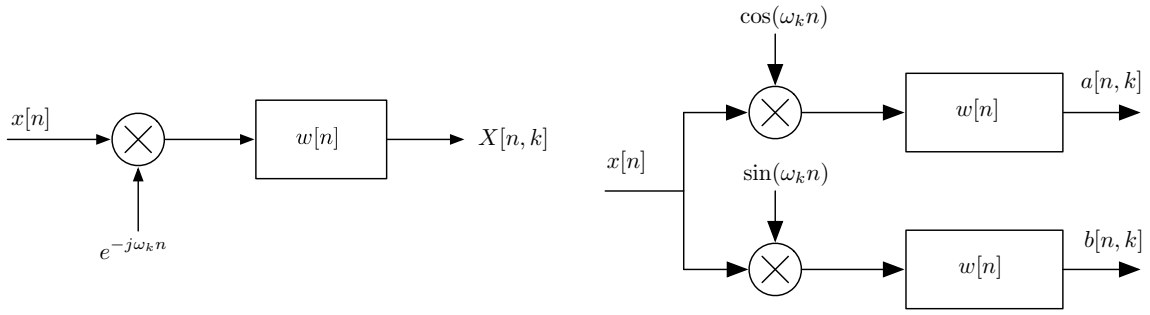

Figure 3.12: The lowpass filter interpretation of the STFT in complex form (left panel) and with real coefficients (right panel)s.

Figure 3.12 recapitulates a single channel of the lowpass filter implementation of short-time Fourier analysis, in both the original complex exponential form (left panel), and using real coefficients (right panel). Note that we are now using a slightly different notation to represent the STFT coefficients as

$$X[n,k] = a[n,k] - jb[n,k] \tag{3.32}$$

where $a[n,k] = Re[X[n,k]]$ and $b[n,k] = -Im[X[n,k]]$.

Now let us consider the STFT coefficients in terms of their magnitude and phase. Specifically, we can represent $X[n,k]$ as

$$X[n,k] = |X[n,k]|e^{j\theta[n,k]} \tag{3.33}$$

where $\theta[n,k]$ represents the instantaneous phase at frame n and frequency bin k. Clearly, we can obtain the magnitude and phase of the STFT directly from the real and imaginary parts $a[n,k]$ and $b[n,k]$ via the standard trigonometric relations:

$$|X[n,k]| = \sqrt{a^2[n,k] + b^2[n,k]} \text{ and } \theta[n,k] = -\tan^{-1}\left(\frac{b[n,k]}{a[n,k]}\right) \tag{3.34}$$

Note that if the STFT coefficients $X[n,k]$ are Hermitian symmetric, by taking the inverse transform we can represent the corresponding time function for that channel as

$$x_k[n] = |X[n,k]|\left(e^{j\theta[n,k]}e^{j\omega n} + e^{-j\theta[n,k]}e^{-j\omega_k n}\right) = 2|X[n,k]|\cos(\omega_k n + \theta[n,k]) \tag{3.35}$$

and in principle the entire waveform could be reconstructed by summing these cosines across all channels:

$$x[n] = \sum_k x_k[n] = \sum_k 2|X[n,k]|\cos(\omega_k n + \theta[n,k]) \tag{3.36}$$

The original idea of phase vocoding was to extract and transmit the magnitude and phase terms representing the signal at each frame, reconstructing the waveform at the far end from this information only.

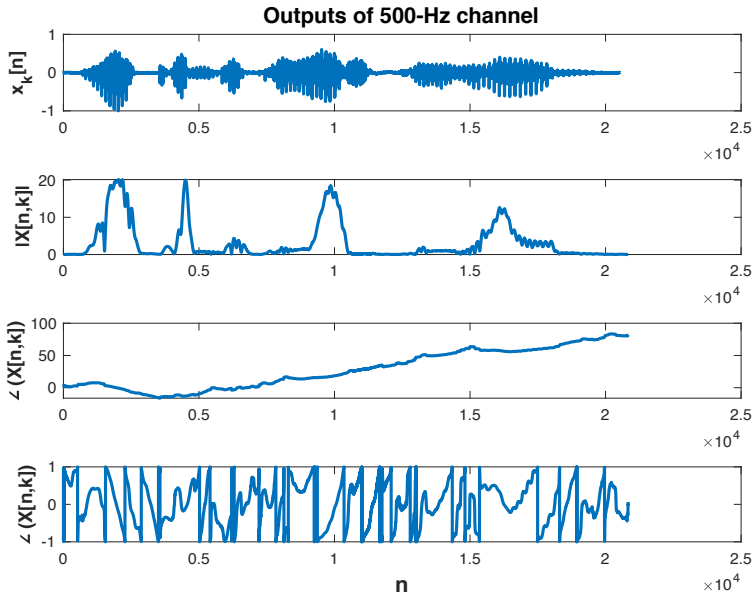

Figure 3.13: Representations obtained via phase vocoding for a channel at $\omega_k = 2\pi 500$. From top to bottom, depicted for that channel are (a) $x_k[n]$, $|X[n,k]|$, $\angle X[n,k]$ (unwrapped), and $\angle X[n,k]$ (wrapped and normalized by dividing by π).

Figure 3.13 depicts the magnitude and phase of the "Welcome to DSP-I" utterance for a typical channel at 500 Hz. The figures show the representation of the waveform in that channel as reconstructed using Eq. (3.35), along with the magnitude and phase of the STFT representation. Note that the phase is presented twice: in its original "unwrapped" form and in terms of the principal value, with the values of $\theta[n,k]$ constrained to lie between $-\pi$ and π radians.

From Fig. 3.13 it can be seen that both forms of the phase function are problematic. Specifically, if the phase is transmitted in its original unwrapped form, the magnitude of the phase is unbounded and in general will get large quickly, ultimately causing overflow after a sufficient time. If only the principal value is transmitted, that signal will have a very large bandwidth because of the abrupt transitions by 2π radians when the magnitude of the phase exceeds π in either direction. The solution to this problem is to transmit an approximation to the *derivative with respect to time* of the phase function.

To simplify the analysis of this approach, we will briefly detour into continuous-time processing. Specifically, let the continuous-time and continuous-frequency representation that is similar to that of Eqs. (3.35) and (3.36) be:

$$x(t) = \sum_k x_k(t) = \sum_k 2|X[t,\Omega]|\cos(\omega_k t + \theta[t,\Omega]) \tag{3.37}$$

where

$$x_k(t) = 2|X(t,\Omega)|\cos(\omega_k t + \theta(t,\Omega)) \tag{3.38}$$

Note that the derivative of the argument of the cosine term, $\omega_k + \dot{\theta}(t,\Omega_k)$, has the dimension of *frequency*. In fact, the derivative of the phase term $\dot{\theta}(t,\Omega_k)$ is considered to be the instantaneous frequency of that channel. One of the motivations for transmitting the magnitude and phase derivative rather than the magnitude and phase is that the instantaneous frequency of a signal typically changes slowly because of physical limitations on how that signal was produced, regardless of whether the source is a human or a machine.

Generalizing, we can define the continuous-time analog of the STFT as

$$X_c(t,\Omega_k) = \int_{-\infty}^{\infty} x_c(\tau) w_c(t-\tau) e^{-j\omega_k \tau} d\tau \tag{3.39}$$

where $w_c(t)$ is a continuous-time window function, and the other variables are in direct correspondence to their discrete-time counterparts. This suggests that the STFT coefficients in continuous time and frequency can be represented as

$$X_c(t,\Omega_k) = |X_c(t,\Omega_k)| e^{j\theta_c(t,\Omega_k)} = a_c(t,\Omega_k) - j b_c(t,\Omega_k) \tag{3.40}$$

where

$$|X_c(t,\Omega_k)| = \left[a_c^2(t,\Omega_k) + b_c^2(t,\Omega_k)\right]^{1/2} \tag{3.41}$$

and

$$\theta_c(t,\Omega_k) = -\tan^{-1}\left[\frac{b_c(t,\omega_k)}{a_c^2(t,\Omega_k)}\right] \tag{3.42}$$

Using the relationship

$$\frac{d}{dx}\tan^{-1}(x) = \frac{1}{1+x^2} \tag{3.43}$$

it follows directly that *instantaneous frequency* can be represented as

$$\dot{\theta}_c(t,\Omega_k) = \frac{b_c(t,\Omega_k)\dot{a_c}(t,\Omega_k) - a_c(t,\Omega_k)\dot{b_c}(t,\Omega_k)}{a_c^2(t,\Omega_k) + b_c^2(t,\Omega_k)} \tag{3.44}$$

The discrete-time analogy to Eq. (3.44) is

$$\dot{\theta}[n,\omega_k] = \frac{b[n,\omega_k]\dot{a}[n,\omega_k] - a[n,\omega_k]\dot{b}[n,\omega_k]}{a^2[n,\omega_k] + b^2[n,\omega_k]} \tag{3.45}$$

These equations have the advantage that they do not require the arctan operation to be evaluated directly.

Figures 3.14 and 3.15 summarize the encoding and decoding steps for phase vocoding. In the encoding process, the magnitude and phase derivative, $|X[n,k]|$ and $\dot{\theta}[n,k]$, are

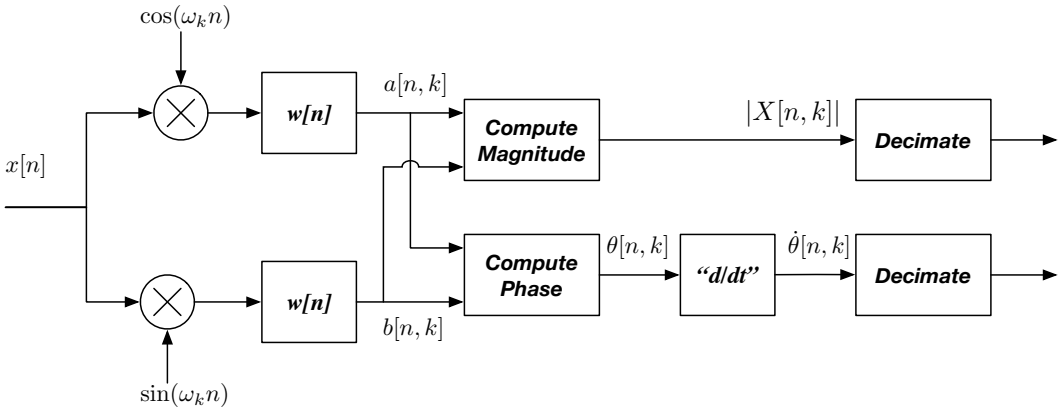

Figure 3.14: Block diagram of the encoding portion of the phase vocoder.

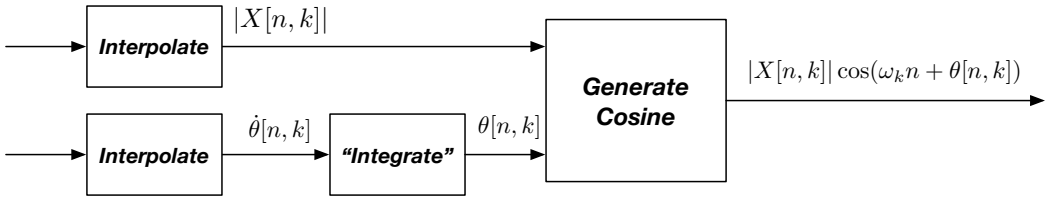

Figure 3.15: Block diagram of the decoding portion of the phase vocoder.

computed from the real and imaginary parts of the STFT coefficients, $a[n,k]$ and $b[n,k]$ according to Eq. (3.34) for the magnitude and Eq. (3.45) for the phase derivative. While the differentiation in time in the expressions $\dot{a}[n,k]$ and $\dot{b}[n,k]$ in Eq. (3.44) cannot be implemented exactly in discrete time, there are many discrete-time approximations to continuous-time differentiation of which the simplest is the *first difference*:

$$x_{\text{diff}}[n] = \frac{1}{T}(x[n] - x[n-1]) \tag{3.46}$$

where T is the sampling period. This is a reasonable approximation to differentiation at low frequencies for which $\sin(\omega n/2) \approx \omega n/2$. In addition, there are many standard filter design techniques (including the Parks-McClellan equiripple filter design method) that produce approximations to ideal differentiation that are valid over a much greater frequency range.

As Fig. 3.15 indicates, conversion from the magnitude and phase-derivative representation to the time function is begun by "integrating" the phase derivative to obtain the instantaneous running phase. This is normally approximated by computing the running cumulative sum of the phase derivative after processing is completed. The instantaneous phase and the magnitude are combined to obtain the time-domain signal in each channel:

$$x_k[n] = 2|X[n,k]|\cos(\omega_k n + \theta[n,k]) \tag{3.47}$$

where as usual $\omega_k = 2\pi k/N$. The signals in each positive frequency channel $x_k[n]$ are summed over all channels to obtain the output waveform.

Time compression and transposition. Two popular uses of phase vocoding are in changing the rate of speech without affecting intelligibility, and in musical transposition. Specif-

ically, multiplying all of the instantaneous frequencies of the representation of a musical performance by a constant will increase the pitch by the same constant. For example, multiplying the instantaneous frequencies by $6/5 = 1.2$ will increase the pitch by the ratio of $6 : 5$, which is equivalent to transposing upward by a minor third in music.

Speech rate can be changed by scaling the instantaneous frequencies by a constant factor while changing the sampling rate by that same factor. For example, *slowing down* speech by a factor of 1.2 is easily accomplished by multiplying the instantaneous frequency by 1.2 while increasing the sampling frequency by the same factor.