Unit 3. Digital Audio Representation and Processing

CS 3570

Shang-Hong Lai

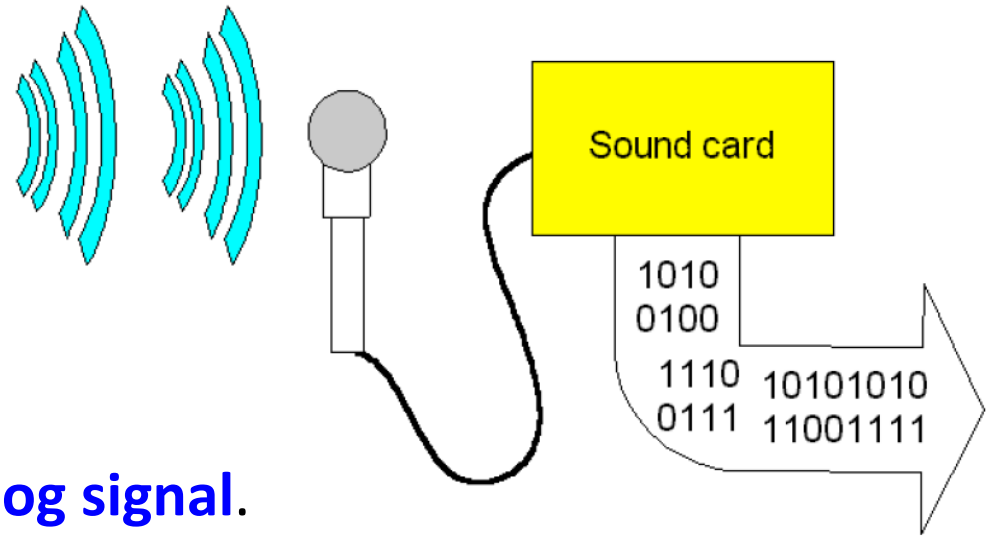
Content

- Audio signal sampling, digitization
- Audio Dithering
- Noise Shaping
- Frequency Analysis
  - Discrete Fourier Transform(DFT)
  - Fast Fourier Transform(FFT)
- Musical acoustics and notation
- Dynamic processing
- Audio restoration
- Digital Audio Filter
  - FIR (finite-impulse response) filter
  - IIR (infinite-impulse response) filter
- Digital Audio Compression
  - MPEG-1 Audio Compression
- Musical Instrument Digital Interface (MIDI)
- Speech Processing
- Music Recommendation

Introduction

- Sound is a mechanical wave that is an oscillation of pressure transmitted through a solid, liquid, or gas.
- The perception of sound in any organism is limited to a certain range of frequencies(20Hz~20000Hz for humans).
- How do we process “sound”?
  - The changing air pressure caused by sound is translated into changing voltages.
  - The fluctuating pressure can be modeled as continuously changing numbers—a function where time is the input variable and amplitude (of air pressure or voltage) is the output.

Digitizing Sound



- Microphone:
  - Receives sound
  - Converts to **analog signal**.
- Computer like **discrete entities**

Need to convert Analog-to-Digital

Hardware (e.g. Soundcard)

Also known as Digital Sampling

Sample Rate and Bit Size

Bit Size — Quantization

- How do we store each sample value (Quantization)?
  - 8 Bit Value (0-255) or (-128~127)
  - 16 Bit Value (Integer) (0-65535) or (-32768~32767)

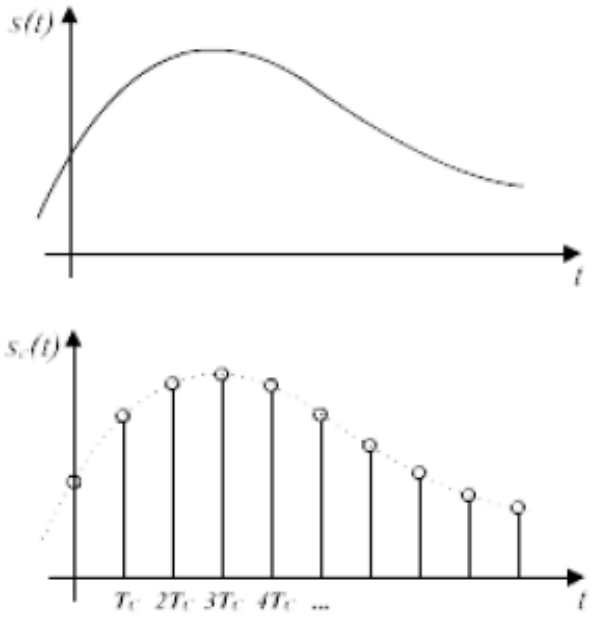
Sample Rate

- How many Samples to take?
  - 8 KHz — Voice quality for phones and toys
  - 16 KHz — Commonly used for speech recognition
  - 44.1 KHz — CD Quality

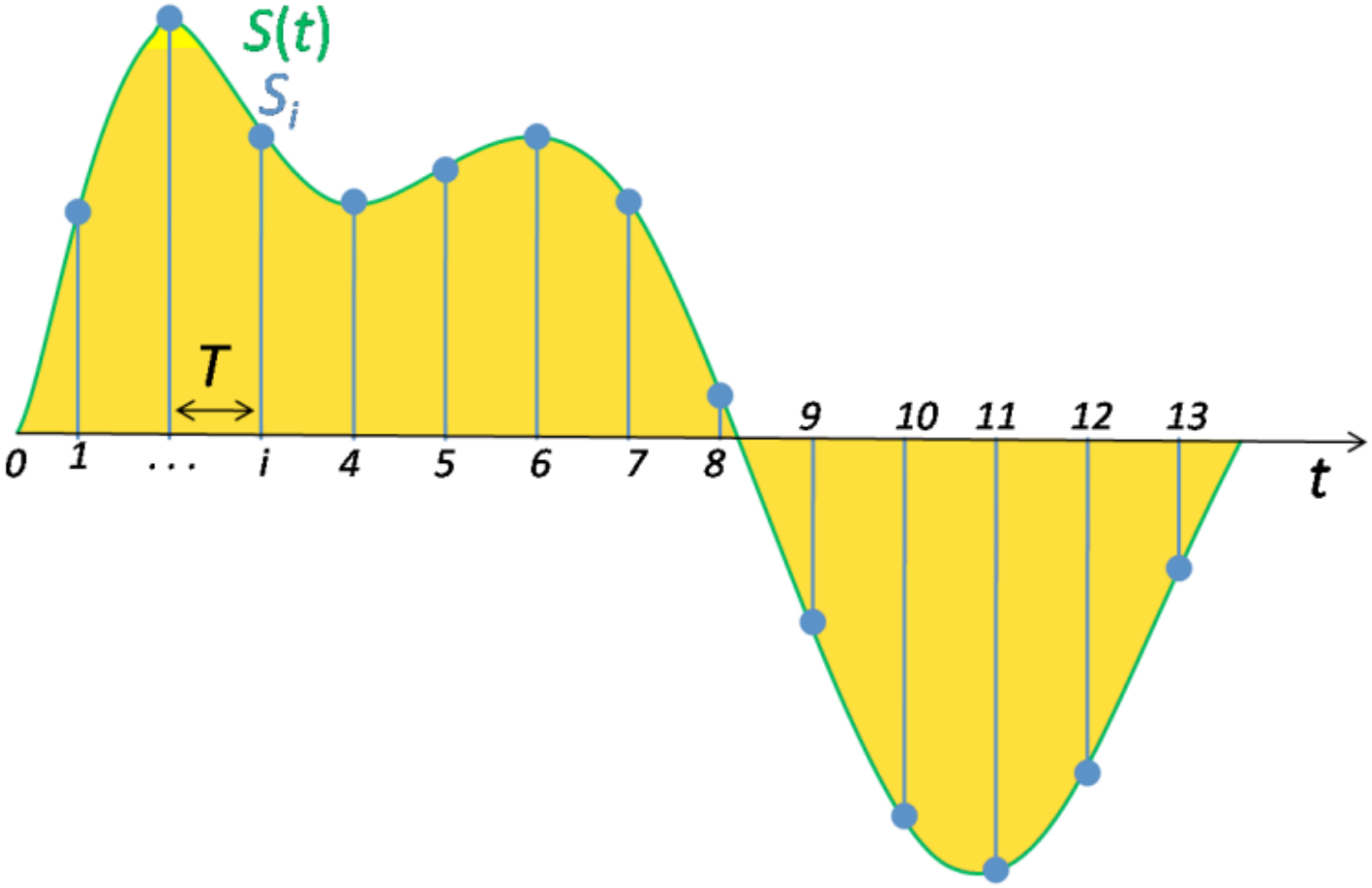
Digital Sampling (1)

Sampling process basically involves:

- Measuring the analog signal at regular discrete intervals
- Recording the values at these points



Digital Sampling (2)



Nyquist Theorem

The Sampling Frequency is critical to the accurate reproduction of a digital version of an analog waveform

Nyquist's Sampling Theorem

The Sampling frequency for a signal must be **at least twice** the **highest frequency component** in the signal.

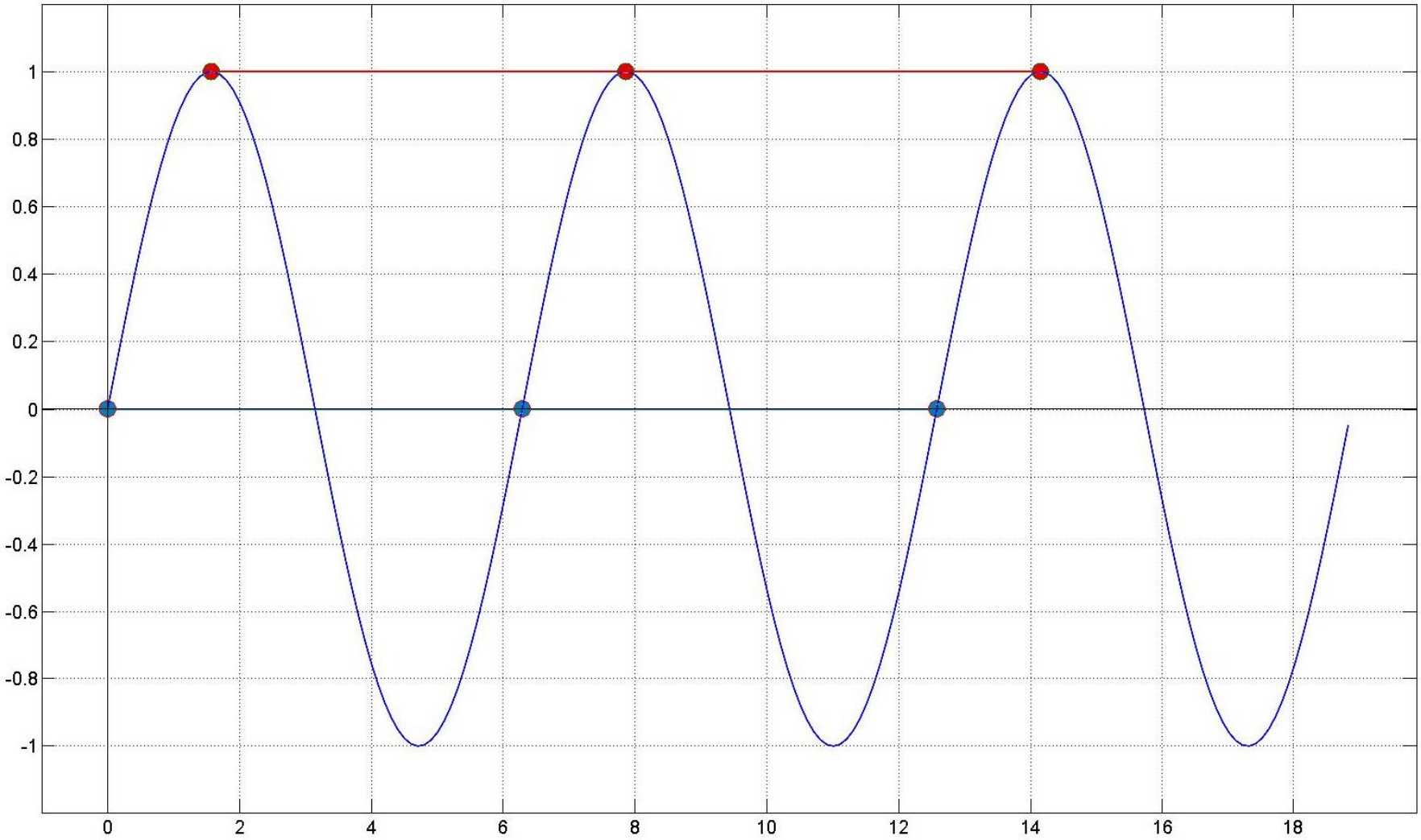
Nyquist Theorem

- Nyquist frequency
  - Given a sampling rate, the ***Nyquist frequency*** is the highest actual frequency component that can be sampled without aliasing.
- Ex:
  - If we choose a sample rate of 8000Hz, the Nyquist frequency $f_{nf} = \frac{1}{2} f_{samp} = 4000\text{Hz}$

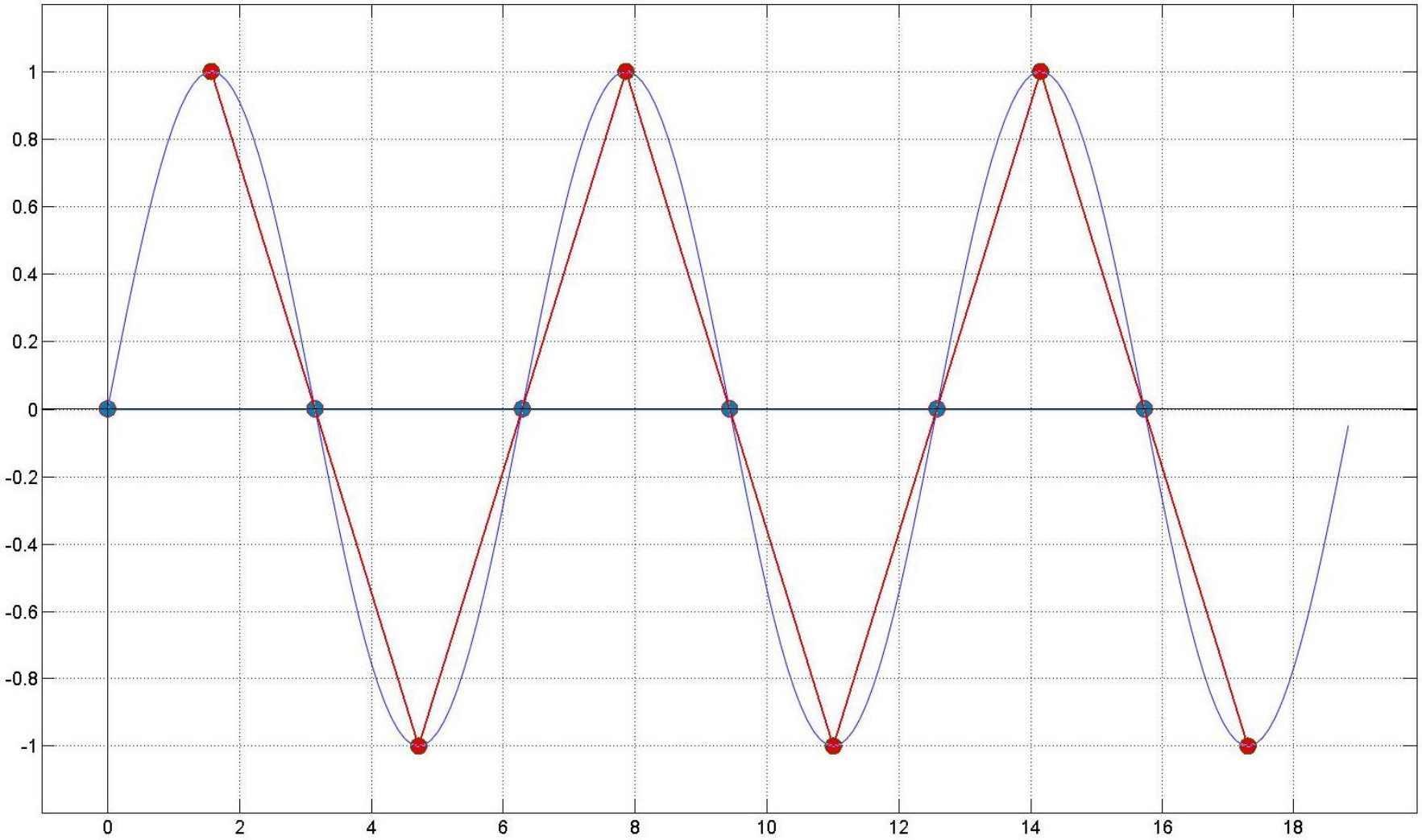
Nyquist Theorem

- Nyquist rate
  - Given an actual frequency to be sampled, the ***Nyquist rate*** is the lowest sampling rate that will permit accurate reconstruction of an analog digital signal.
- Ex:
  - If the highest frequency component is 10,000Hz, the Nyquist rate $f_{nr} = 2f_{max} = 20{,}000\text{Hz}$

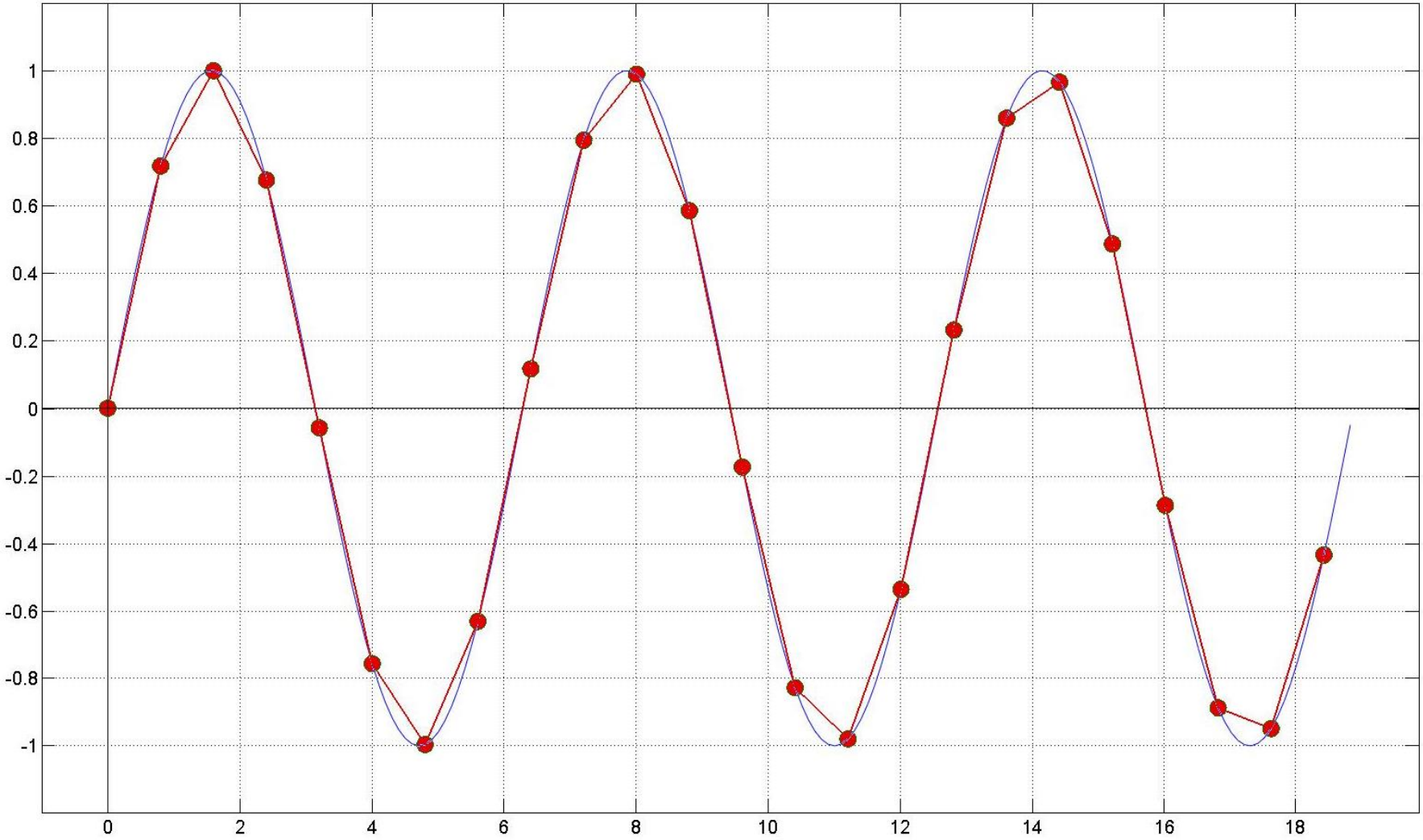
Sampling at Signal Frequency



Sampling at Twice Nyquist Frequency

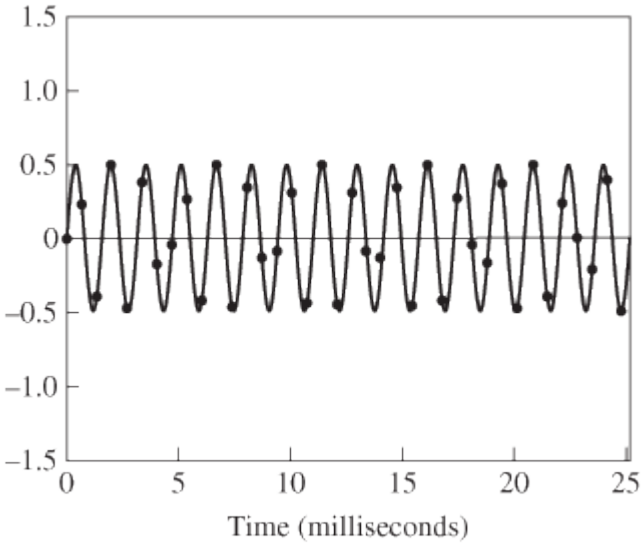


Sampling above Nyquist Rate



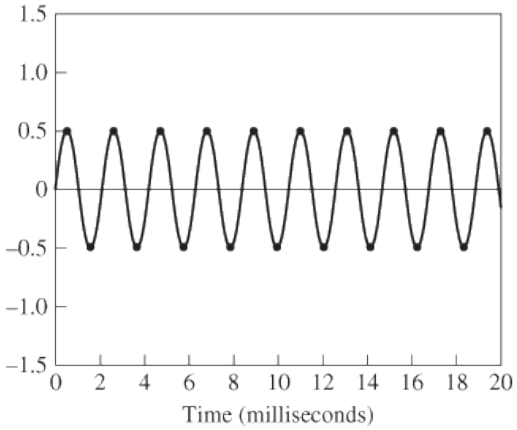
Sampling Rate and Aliasing

- In essence, the reason a too-low sampling rate results in aliasing is that there aren't enough sample points from which to accurately interpolate the sinusoidal form of the original wave.

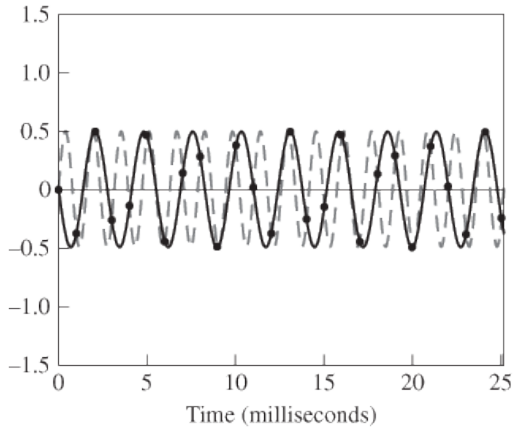


Samples taken more than twice per cycle will provide sufficient information to reproduce the wave with no aliasing

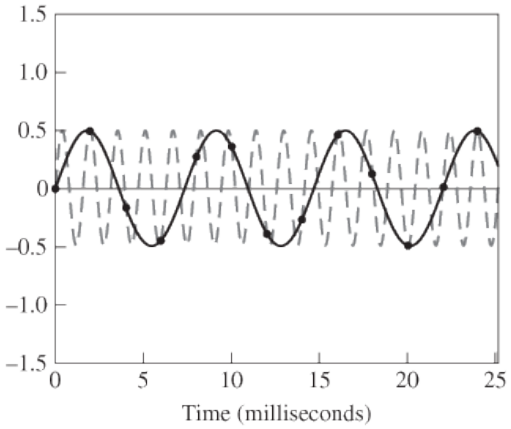
Sampling Rate and Aliasing



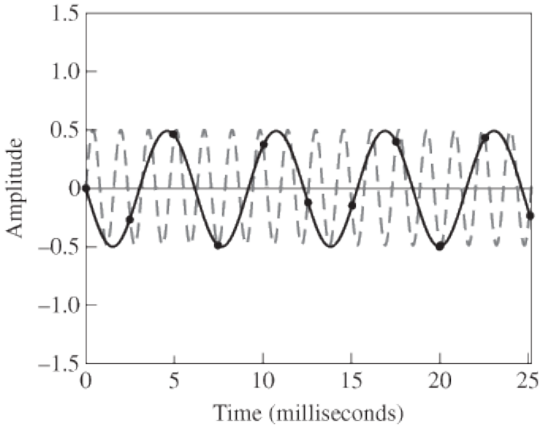
Samples taken exactly twice per cycle *can* be sufficient for digitizing the original with no aliasing



A 637 Hz wave sampled at 1000 Hz aliases to 363 Hz



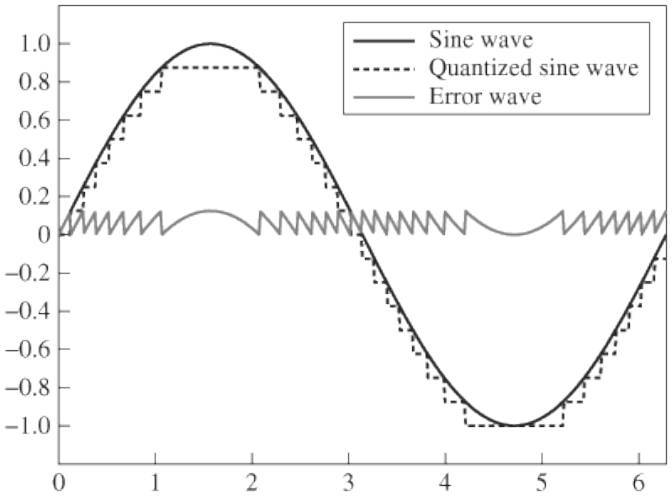
A 637 Hz wave sampled at 500 Hz aliases to 137 Hz



A 637 Hz wave sampled at 400 Hz aliases to 163 Hz

Audio Dithering

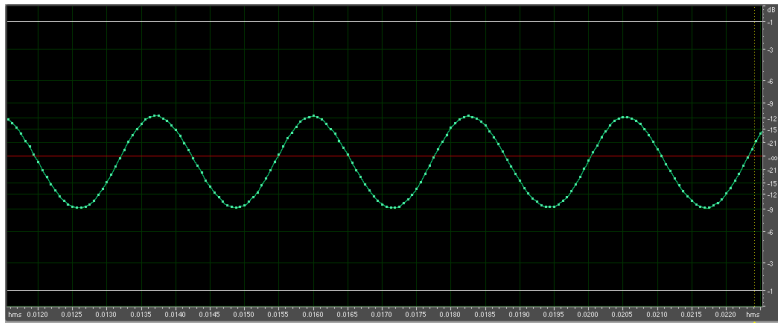
- ***Audio dithering*** is a way to compensate for quantization error.
- Quantized signals would sound 'granular' because of the stair-step effect. The quantized signals sound like the original signals plus the noise.
- The noise follows the same pattern as the original wave, human ear mistakes it as the original signal.



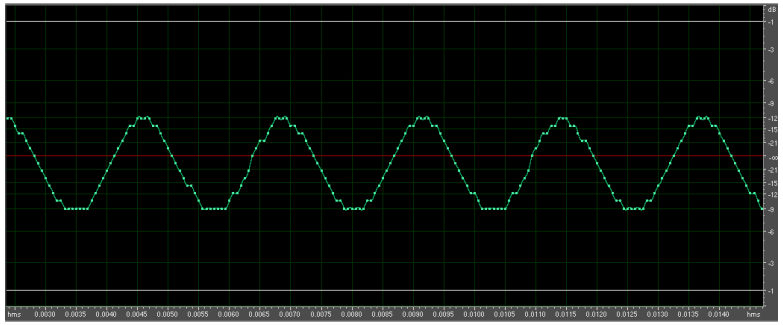
Audio Dithering

- Example1-simpe wave

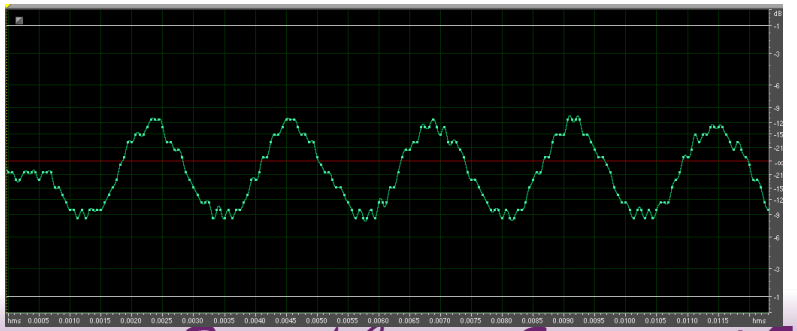
Original wave



After bit reduction

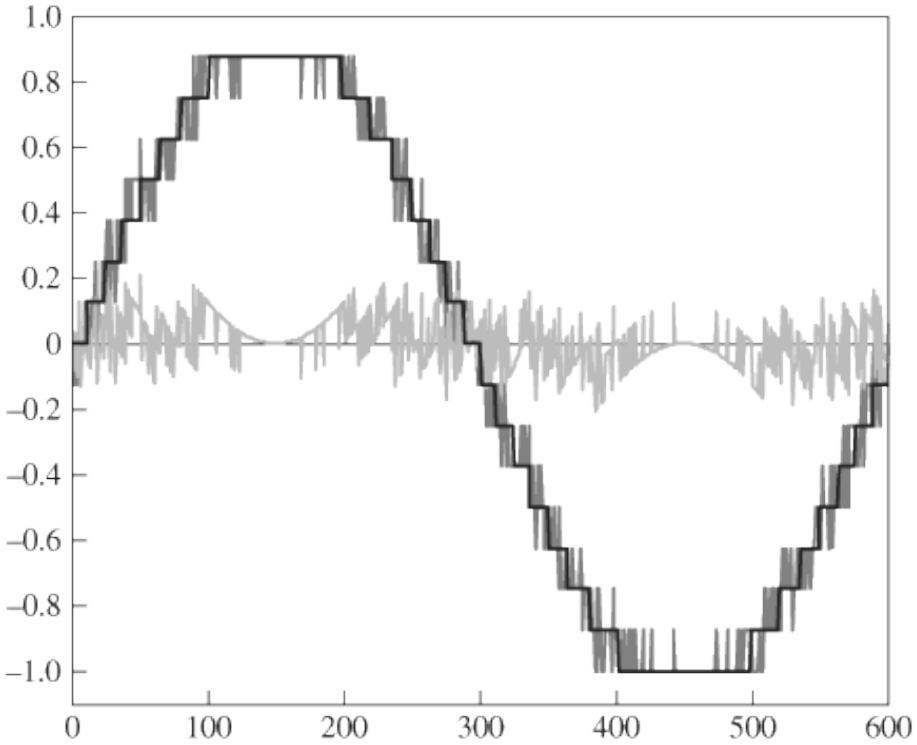


After dithering



Audio Dithering

- Adding a random noise (dither) to the original wave eliminates the sharp stair-step effect in the quantized signal.
- The noise is still there, but has less effect on the original signal.(we can hear the smooth signal without stair-step effect)

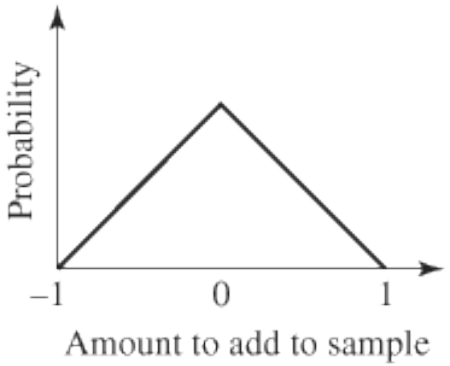


Dithered quantized wave

Audio Dithering

- Dithering function
  - Triangular probability density function (TPDF)

Triangular probability function



  - Rectangular probability density function (RPDF): All numbers in the selected range have the same probability
  - Gaussian PDF: The Gaussian PDF weights the probabilities according to a Gaussian
  - Colored dithering: Colored dithering produces noise that is not random and is primarily in higher frequencies.

Audio Dithering

- Example2 – complex wave

Original wave 

After bit reduction 

After dithering 

Noise Shaping

- ***Noise shaping*** is another way to compensate for the quantization error. Noise shaping is *not* dithering, but it is often used along with dithering.

- The idea behind noise shaping is to redistribute the quantization error so that the noise is concentrated in the higher frequencies, where human hearing is less sensitive, or we can use a low-pass filter to filter out the high frequency components.

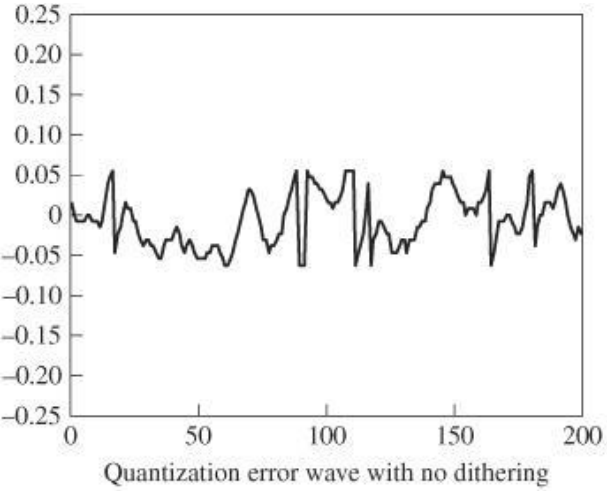
Noise Shaping

- First-order feedback loop for noise shaping
  - Let ***F_in*** be an array of N digital audio samples that are to be quantized, dithered, and noise shaped, yielding ***F_out***. For $0 \le i \le N - 1$, define the following: $\boldsymbol{F\_in}_i$ is the ith sample value, not yet quantized.
  - $\boldsymbol{D}_i$ is a random dithering value added to the ith sample.
  - The assignment statement $\boldsymbol{F\_in}_i = \boldsymbol{F\_in}_i + \boldsymbol{D}_i + c\boldsymbol{E}_{i-1}$ dithers and noise shapes the sample. Subsequently, $\boldsymbol{F\_out}_i = [\boldsymbol{F\_in}_i]$ quantizes the sample.
  - $\boldsymbol{E}_i$ is the error resulting from quantizing the ith sample after dithering and noise shaping.
  - For $i = -1$, $\boldsymbol{E}_i = 0$. Otherwise, $\boldsymbol{E}_i = \boldsymbol{F\_in}_i - \boldsymbol{F\_out}_i$.

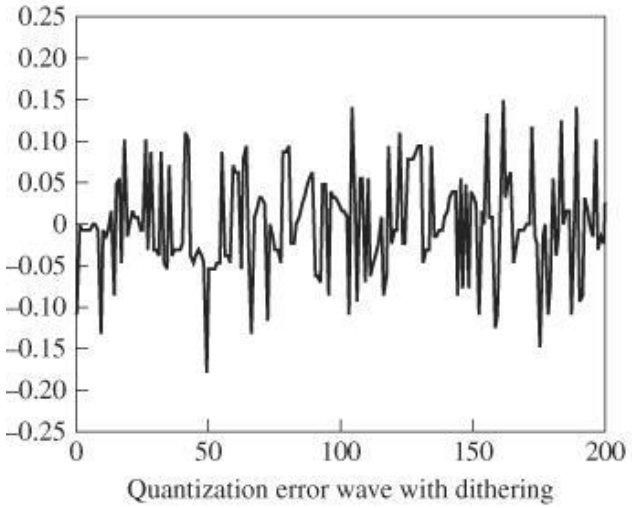
Noise Shaping

- What does noise shaping do?
  - Move noise's frequency to above the Nyquist frequency, and filter it out. We are not losing anything we care about in the sound.
- The term *shaping* is used because you can manipulate the "shape" of the noise by manipulating the noise shaping equations
- The general statement for an *n*-th order noise shaper noise shaping equation becomes $\boldsymbol{F\_out}_i = \boldsymbol{F\_in}_i + \boldsymbol{D}_i + c_{i-1}\boldsymbol{E}_{i-1} + c_{i-2}\boldsymbol{E}_{i-2} + \cdots + c_{i-n}\boldsymbol{E}_{i-n}$.

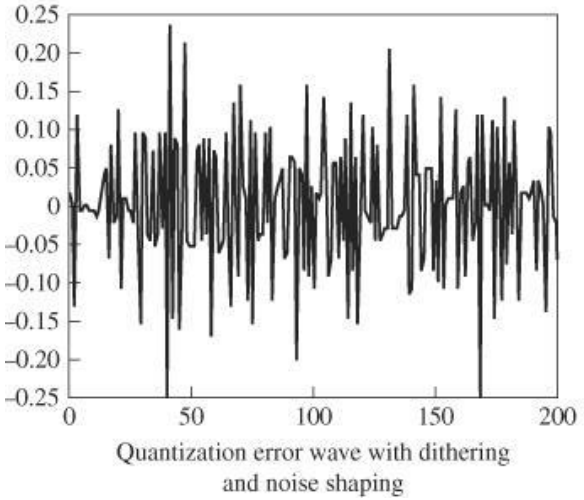
Noise Shaping



No dithering



With dithering



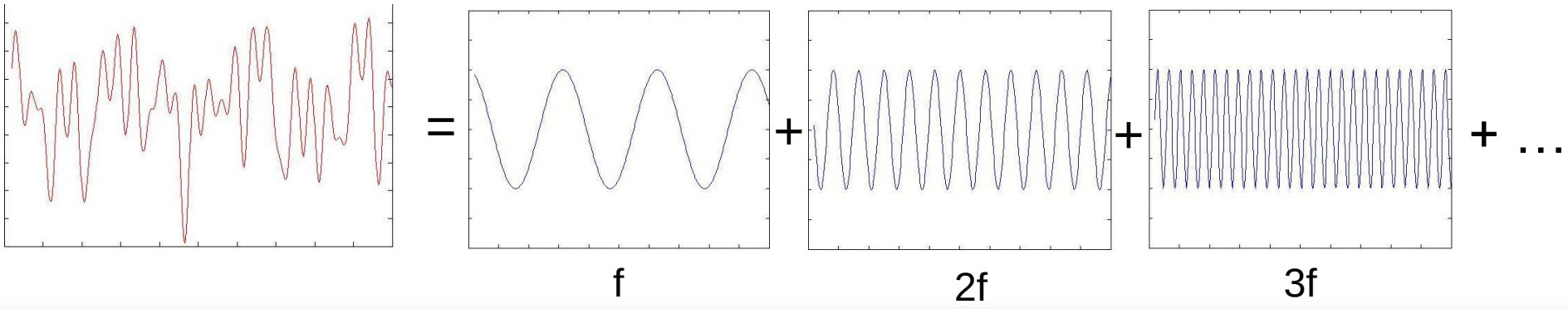
With dithering
and noise shaping

Noise Shaping

- Original 
- After bit reduction 
- After dithering 
- Dithering with noise shaping 

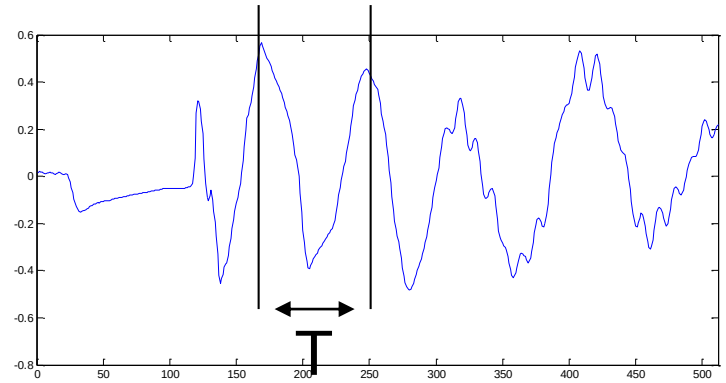
Frequency Analysis

- Time domain
  - Input: time (x-axis)
  - Output: amplitude(y-axis)
- A complex waveform is equal to an infinite sum of simple sinusoidal waves, beginning with a ***fundamental frequency*** and going through frequencies that are integer multiples of the fundamental frequency – ***harmonic frequencies***.

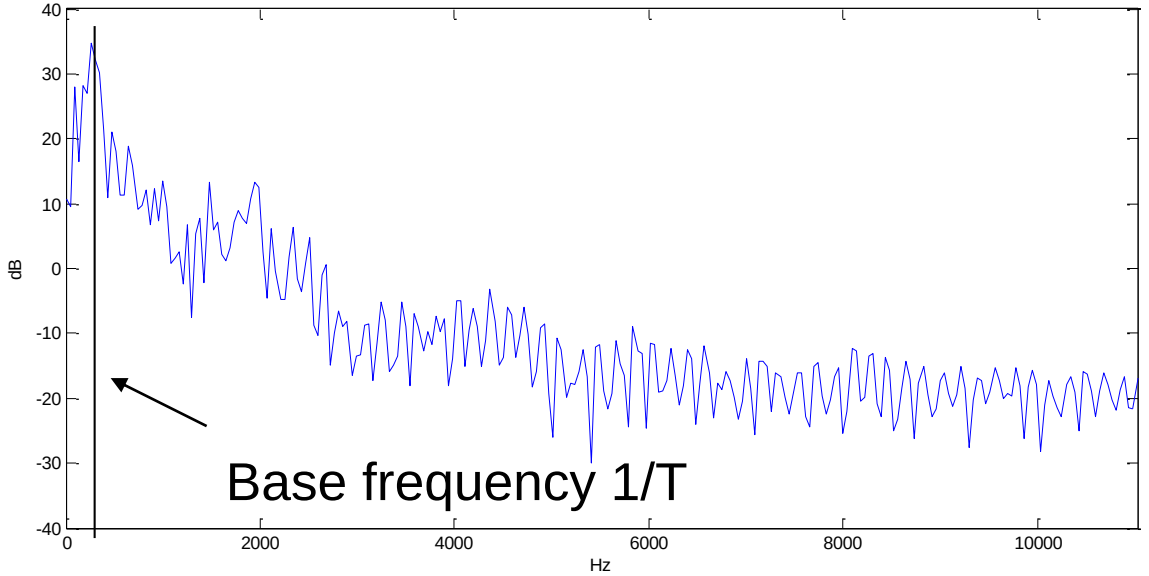


Frequency Domain Analysis

- **Fourier transform** can be used to decompose any signal into summation of sinusoidal waves.
- In Matlab, we can use **fft** (Fast Fourier Transform) for frequency domain analysis.



The time domain waveform



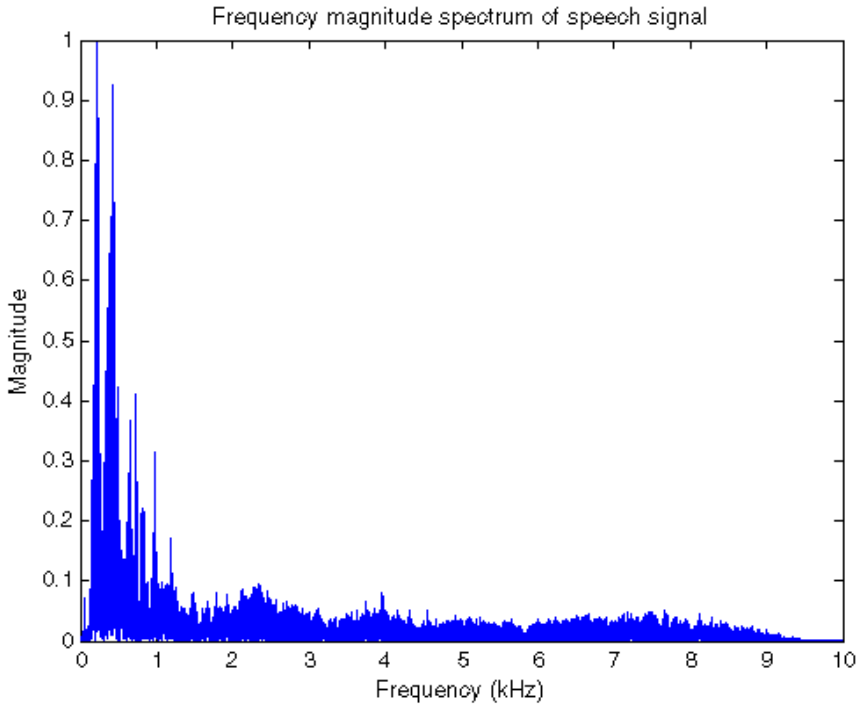
The frequency domain components.

Frequency Analysis

- Two views for frequency analysis
  - Frequency analysis view(spectrum analysis view)- common

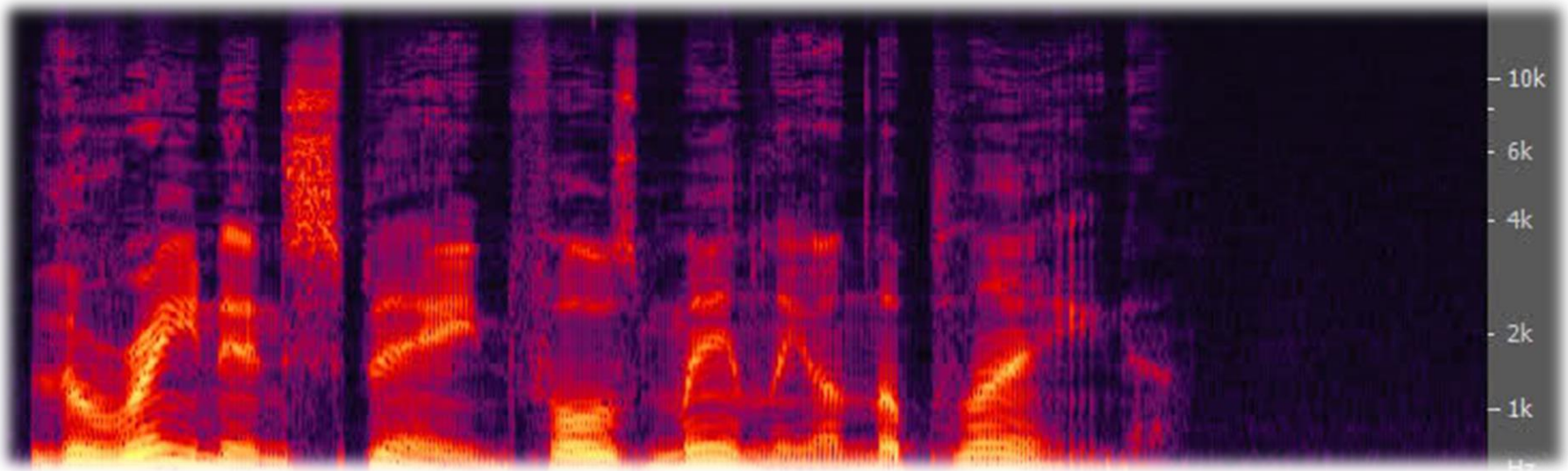
    x-axis: frequency

    y-axis: magnitude of the frequency component



Frequency Analysis

- Spectral view
  - x-axis: time, y-axis: frequency
  - color: magnitude of the frequency component



Discrete Fourier Transform

- The ***discrete Fourier transform*** (**DFT**) operates on an array of *N* audio samples, returning cosine and sine coefficients that represent the audio data in the frequency domain.

$$F_n = \frac{1}{N}\sum_{k=0}^{N-1} f_k \cos\left(\frac{2\pi nk}{N}\right) - i f_k \sin\left(\frac{2\pi nk}{N}\right)$$
$$= \frac{1}{N}\sum_{k=0}^{N-1} f_k e^{\frac{-i2\pi nk}{N}} \qquad (4.8)$$

Discrete Fourier Transform

- Let $\boldsymbol{f}_k$ be a discrete integer function representing a digitized audio signal in the time domain, and $\boldsymbol{F}_n$ be a discrete, complex number function representing a digital audio signal in the frequency domain. Then the ***inverse discrete Fourier transform*** is defined by

$$\begin{aligned} \boldsymbol{f}_k &= \sum_{n=0}^{N-1} \left[\boldsymbol{a}_n \cos\left(\frac{2\pi nk}{N}\right) + \boldsymbol{b}_n \sin\left(\frac{2\pi nk}{N}\right) \right] \\ &= \sum_{n=0}^{N-1} F_n e^{\frac{i2\pi nk}{N}} \end{aligned} \tag{4.7}$$

- Subscript k: signal value at time k

  Subscript n: n^{th} frequency component

Discrete Fourier Transform

- The DC component, $\boldsymbol{a}_0$, is defined by $\boldsymbol{a}_0 = \frac{1}{N}\sum_{k=0}^{N-1} \boldsymbol{f}_k$, giving the average amplitude.
- The AC components are, for 1≤n≤N,

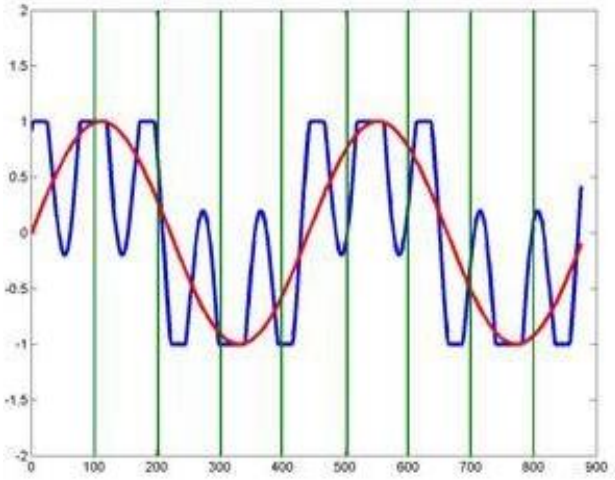
$$\boldsymbol{a}_n = \frac{1}{N}\sum_{k=0}^{N-1} \boldsymbol{f}_k \cos\left(\frac{2\pi nk}{N}\right)$$

$$\boldsymbol{b}_n = \frac{1}{N}\sum_{k=0}^{N-1} \boldsymbol{f}_k sin\left(\frac{2\pi nk}{N}\right)$$

- Fundamental frequency $f = \frac{1}{N}$
- Fundamental angular frequency $\omega = 2\pi f = 2\pi/N$

How does DFT Work?

- Suppose the blue wave represents the complex audio and the red one is a sinusoidal wave of a certain frequency n.
- Green line means the sample points, N=8.
- If the sinusoidal wave fits the signal well, then $\mathbf{F}_n$ is large, which means this frequency component takes a big ratio in the complex signal.



0.9872 * 0.9999	=	0.9871	
0.3015 * 0.7612	=	0.2295	
-0.8994 * -1	=	0.8994	
-0.5633 * -1	=	0.5633	
0.7355 * -0.1226	=	-0.0902	+ → 2.8118
0.7773 * 0.2188	=	0.1700	
-0.5092 * -0.2729	=	0.1390	
-0.9255 * 0.0932	=	-0.0863	

Comparison between DCT and DFT

- You may think now that the DCT is inherently superior to the DFT because it doesn't trouble you with complex numbers, and it yields twice the number of frequency components. But how about the phase information?

- The DFT contains both real part and imaginary part, and thus we can get the phase information. The DCT, however, cancels out the sine terms, together with the phase information.

Phase Information in DFT

- Let the equation for the inverse discrete Fourier transform be as given in (4.7, p31). Then the ***magnitude of the nth frequency component***, $\boldsymbol{A_n}$, is given by

$$\boldsymbol{A_n} = \sqrt{\boldsymbol{a_n^2} + \boldsymbol{b_n^2}} \ , \quad 0 \le n \le N-1$$

- The ***phase of the nth frequency component***, $\emptyset_n$, is given by

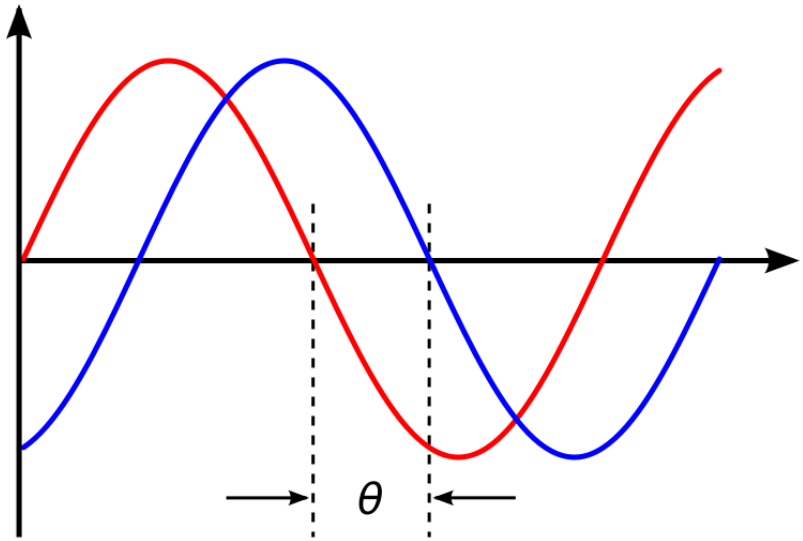
$$\emptyset_n = -tan^{-1}\left(\boldsymbol{b_n}/\boldsymbol{a_n}\right), 0 \le n \le N-1$$

- The ***magnitude/phase form of the inverse DFT*** is given by

$$\boldsymbol{f}_k = \sum_{n=0}^{N-1} \boldsymbol{A}_n \cos(2\pi nk + \emptyset_n)$$

Phase Information

- Phase information in audio
  - Audio is a wave that continuously comes into your ears, so actually we don't detect the phase difference.
  - However, if two waves of the same frequency, one is phase shifted, come to you at the same time, then you will hear the destructive interference.



same frequency
different phase

Fast Fourier Transform(FFT)

- The usefulness of the discrete Fourier transform was extended greatly when a fast version was invented by Cooley and Tukey in 1965. This implementation, called the ***fast Fourier transform* (*FFT*)**, reduces the computational complexity from $O(N^2)$ to $O(N \log_2(N))$. N is the number of samples.

- The FFT is efficient because redundant or unnecessary computations are eliminated. For example, there's no need to perform a multiplication with a term that contains sin(0) or cos(0).

FFT

- What would happen if the window size doesn't fit an integer-multiple of the signal's period?
- For example, Assume that the FFT is operating on 1024 samples of a 440 Hz wave sampled at 8000 samples per second. Then the window contains (1024/8000)*440=56.32 cycles => the end of the window would break the wave in the middle of a cycle.
- Due to this phenomenon(called ***spectral leakage***), the FFT may assume the original signal looks like Fig.4.21.

  The signal becomes discontinuous.

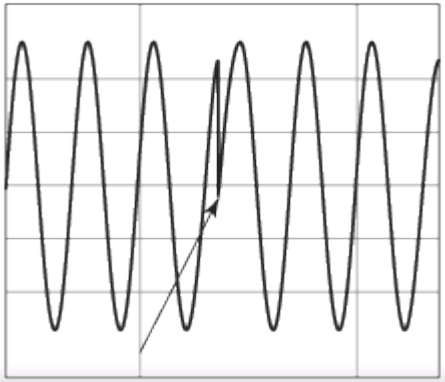
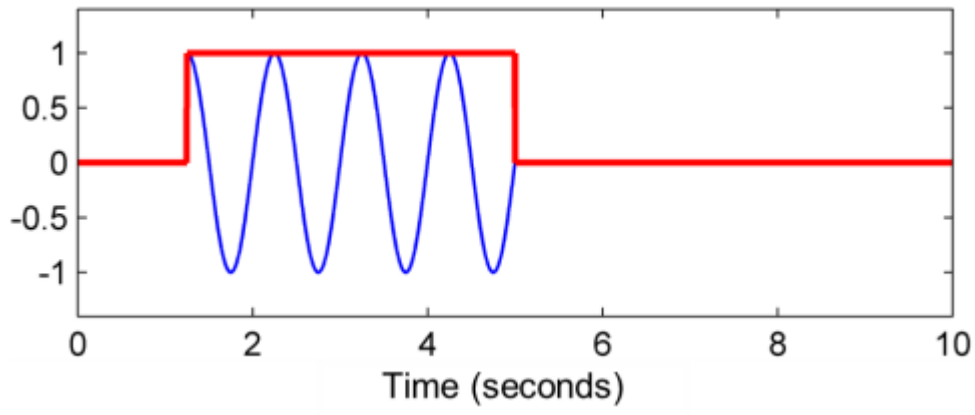
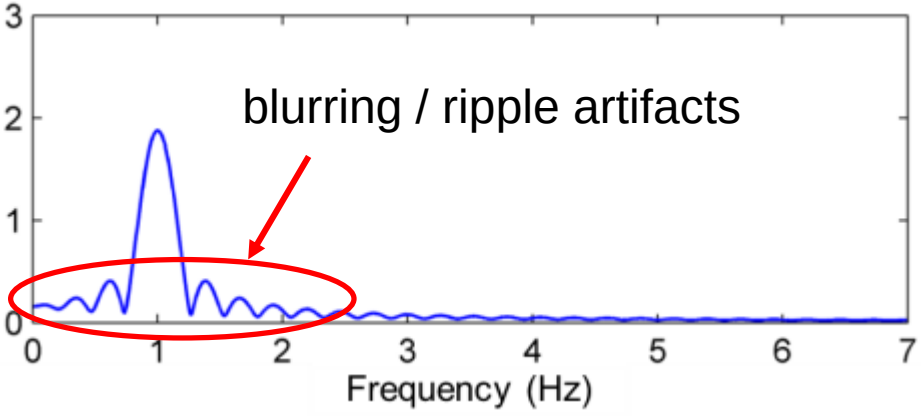


Fig.4.21

Spectral Leakage

- Measure a signal of infinite length is impossible
- Perform Fourier transform on finite samples / time range (Windowing)
- Spectral leakage results in a relatively localized spreading of frequency components, with usually blurring / ripple artifacts





Windowing Function

- **Window function** - to reduce the amplitude of the sound wave at the beginning and end of the FFT window. If the amplitude of the wave is smaller at the beginning and end of the window, then the spurious frequencies will be smaller in magnitude as well.

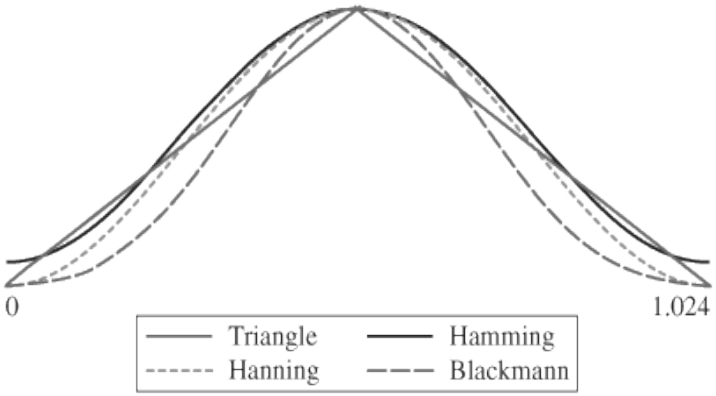


TABLE 4.4 Windowing Function for FFT

$u(t) = \begin{cases} \frac{2t}{T} & for \quad 0 \le t < \frac{T}{2} \\ 2 - \frac{2t}{T} & for \quad \frac{T}{2} \le t \le T \end{cases}$ Triangular windowing function	$u(t) = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi t}{T}\right)\right] \quad for \quad 0 \le t \le T$ Hanning windowing function
$u(t) = 0.54 - 0.46\cos\left(\frac{2\pi t}{T}\right) \quad for \quad 0 \le t \le T$ Hamming windowing function	$u(t) = 0.42 - 0.5\cos\left(\frac{2\pi t}{T}\right) + 0.08\cos\left(\frac{4\pi t}{T}\right)$ for $0 \le t \le T$ Blackmann windowing function

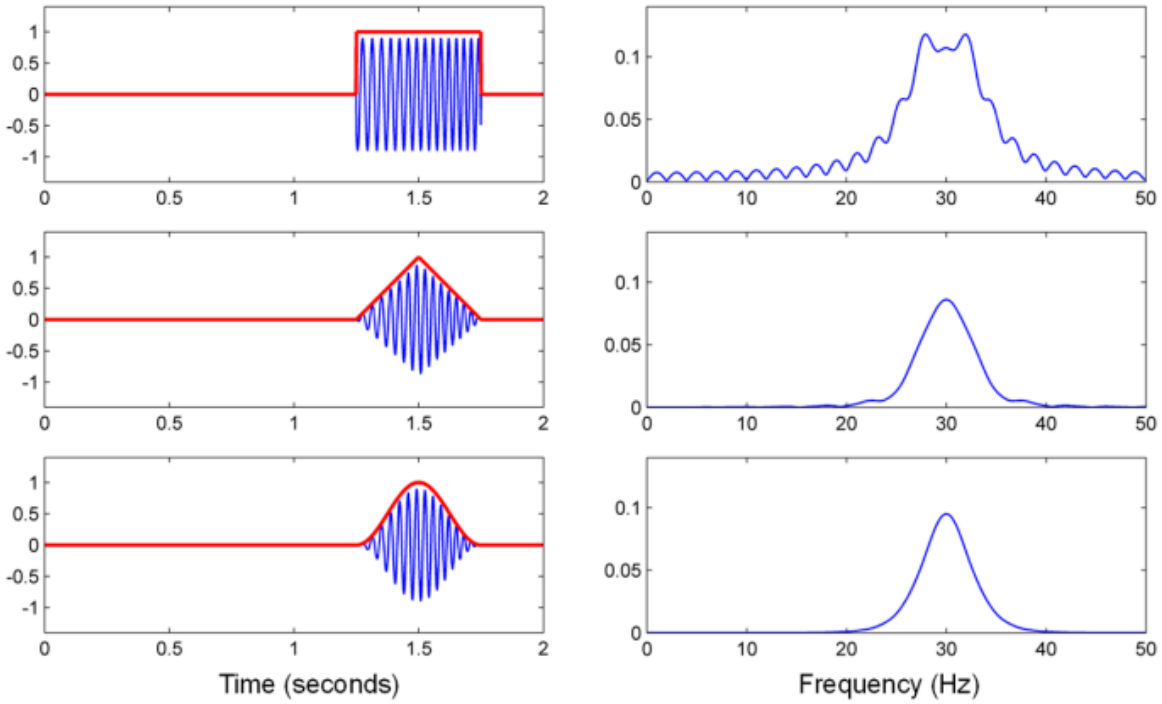
Windowing Function

- **Window function** - to reduce the amplitude of the sound wave at the beginning and end of the FFT window. If the amplitude of the wave is smaller at the beginning and end of the window, then the spurious frequencies will be smaller in magnitude as well.

Rectangular Window

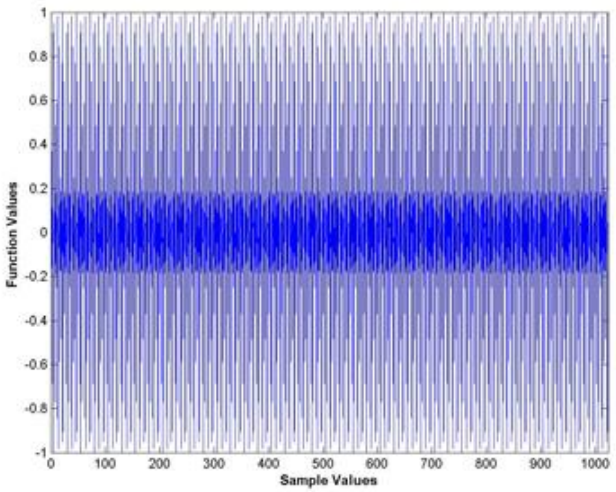
Triangular Window

Hann Window

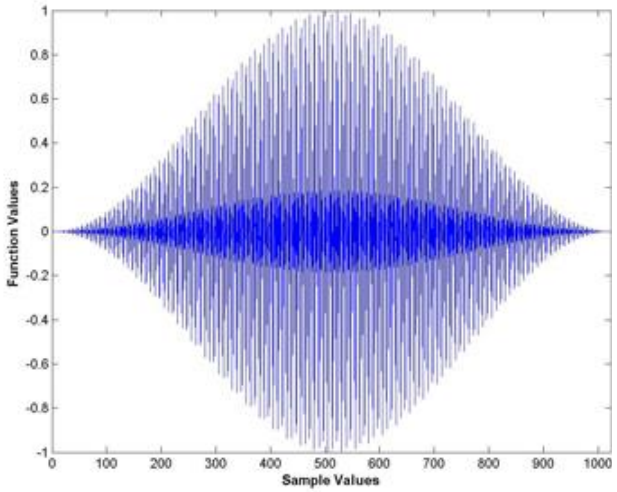


Window Function

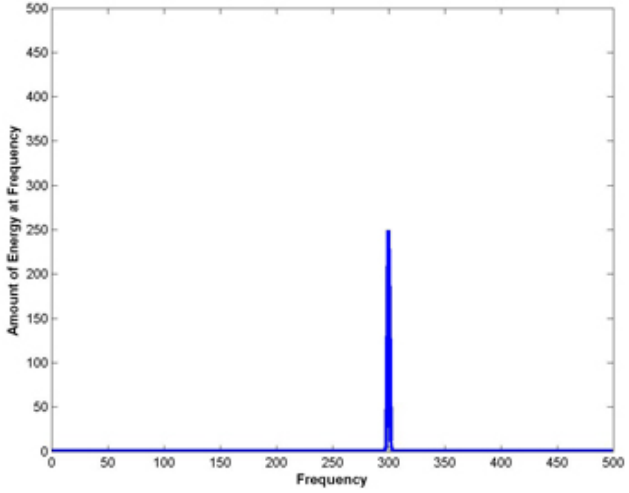
- The frequency components become more accurate, but the magnitudes also decrease. To counteract this, some other algorithms would be used.



Original wave



Applying window function



FFT result

Application of Frequency Analysis

- Translating an audio signal from time domain to frequency domain makes us observe distribution and range of its frequency more directly, then we can do audio processing and recognition, or design different filters.

- Speaker Recognition (Security system)
- Speech imitation (Tom Cat)
- Sound simulation (Animal sounds)

Musical Acoustics and Notation

- The range of human hearing is from about 20 Hz to about 20,000 Hz. As you get older, you lose your ability to hear high frequency sounds.
  - Test if you can hear the frequency you should be able to hear at your age
  - http://www.ultrasonic-ringtones.com/
- Octave equivalence
  - If the frequency of one note is 2^n times of the frequency of another, where *n* is an integer, the two notes sound "the same" to the human ear, except that the first is higher-pitched than the second.(n=1 => 高八度)

Decibels

- Decibels (E_0, I_0: threshold of human hearing)
  - Decibels-sound-pressure-level (dB_SPL)

    $dB\_SPL = 20\log_{10}\left(\frac{E}{E_0}\right), E_0 = 2 \times 10^{-5} Pa$

  - Decibels-sound-intensity-level (dB_SIL)

    $dB\_SIL = 10\log_{10}\left(\frac{I}{I_0}\right), I_0 = 10^{-12} W/m^2$

- Decibels can be used to measure many things in physics, optics, electronics, and signal processing.
- A decibel is **not** an absolute unit of measurement.

DBFS

- In audio processing software, the amplitude is often shown in **dBFS. (*decibels-full-scale*)**

$$\mathrm{dBFS} = 20\log_{10}\left(\frac{Sample\ Level}{Max\ Level}\right)$$

- 0 dBFS represent the highest possible level of sound produced by audio. All other levels are expressed in **negative** numbers.
- dB vs dBFS
  - dB(decibels) are used to describe differences or changes in **sound level**
  - dBFS are used to describe signal levels in comparison with the **highest level your system** can handle.

Dynamic Range

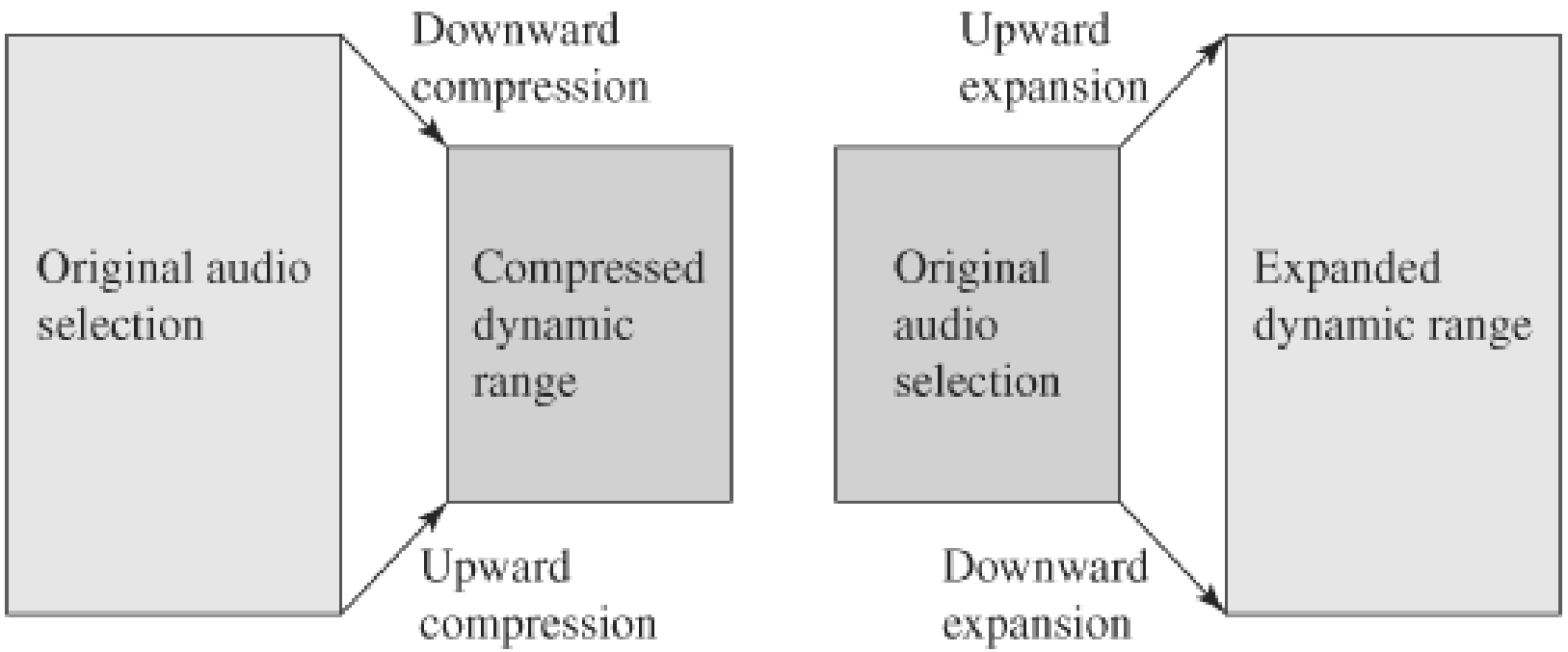
- Dynamic range is the ratio between the smallest nonzero value, which is 1, and the largest, which is 2^n. The *dynamic range of the audio file*, *d*, in decibels, is defined as $d = 20 \log_{10} 2^n = 20n \, log_{10} 2 \approx 6n$
- The definition is identical to the definition of SQNR, and this is why you see the terms SQNR and dynamic range sometimes used interchangeably.
- Be careful not to interpret this as a 16-bit file allows louder amplitudes than an 8-bit file. Rather, dynamic range gives you a measure of the range of amplitudes that can be captured relative to the loss of fidelity compared to the original sound.

Dynamics Processing

- **Dynamics processing** is the process of adjusting the dynamic range of an audio selection, either to reduce or to increase.
- An increase in amplitude is called **gain** or **boost**. A decrease in amplitude is called **attenuation** or, informally, a **cut**.
- We introduce 4 digital dynamics processing tools here: **audio limiting**, **normalization**, **compression**, and **expansion**.

Compression and Expansion

- Types of dynamic range compression and expansion

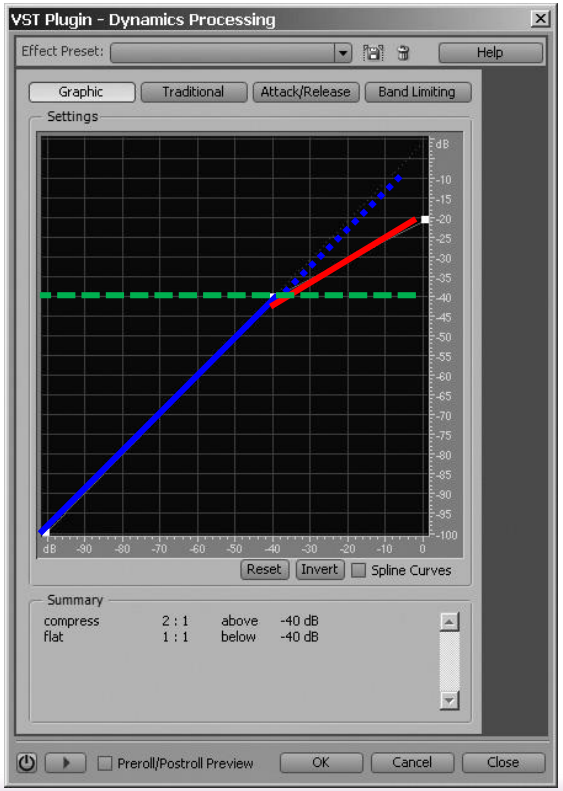


Compression and Expansion

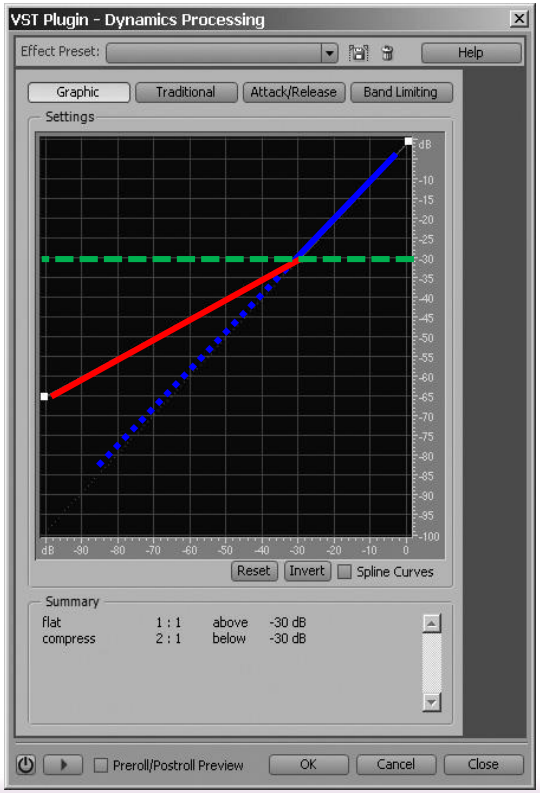
- ***Downward compression*** lowers the amplitude of signals that are above a designated level, without changing the amplitude of signals below the designated level. It reduces the dynamic range.
- ***Upward compression*** raises the amplitude of signals that are below a designated level without altering the amplitude of signals above the designated level. It reduces the dynamic range.
- ***Upward expansion*** raises the amplitude of signals that are above a designated level, without changing the amplitude of signals below that level. It increases the dynamic range.
- ***Downward expansion*** lowers the amplitude of signals that are below a designated level without changing the amplitude of signals above this level. It increases the dynamic range.

Compression and Expansion - Examples

- Downward compression:
- Amplitudes above -40dB is lowered by a 2 : 1 ratio.



- Upward compression:
- Amplitudes below -30dB is compressed by a 2 : 1 ratio.



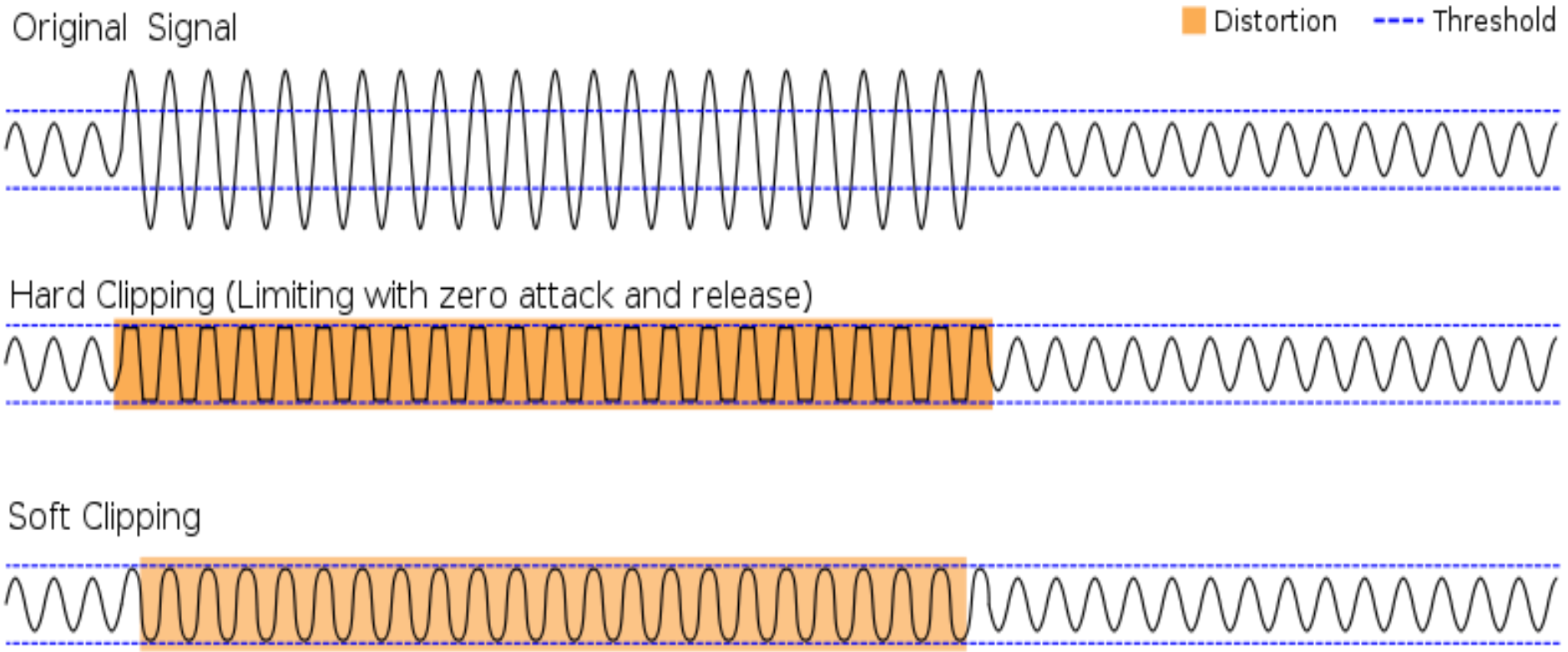
Limiting

- **Audio limiting** limits the amplitude of an audio signal to a designated level.

- Hard limiting (clipping)
  - cuts amplitudes of samples to a given maximum and/or minimum level.

- Soft limiting
  - audio signals above the designated amplitude are recorded at lower amplitude.

Limiting



http://en.wikipedia.org/wiki/File:Clipping_compared_to_limiting.svg

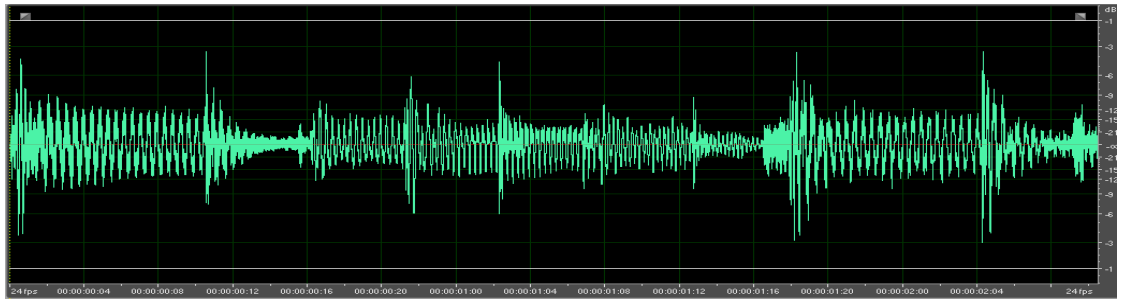
Normalization

- Often, normalization is used to increase the perceived loudness of a piece after the dynamic range of the piece has been compressed.

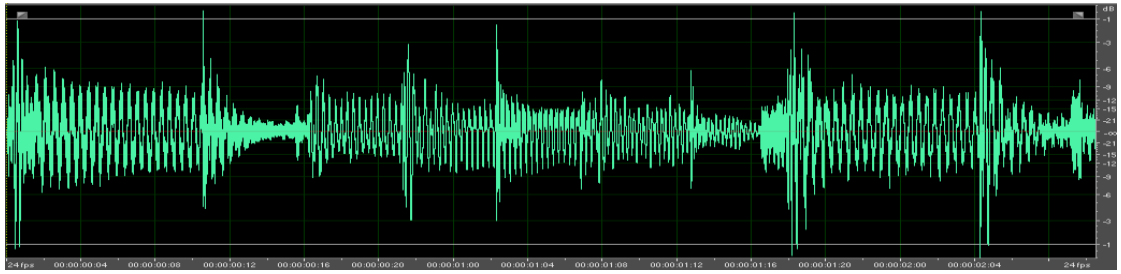
- Normalization steps:
  1. Find the highest amplitude sample in the audio selection.
  2. Determine the gain needed in the amplitude to raise the highest amplitude to maximum amplitude.
  3. Raise all samples in the selection by this amount.

Dynamics Processing - Example

- Bossa.wav - original

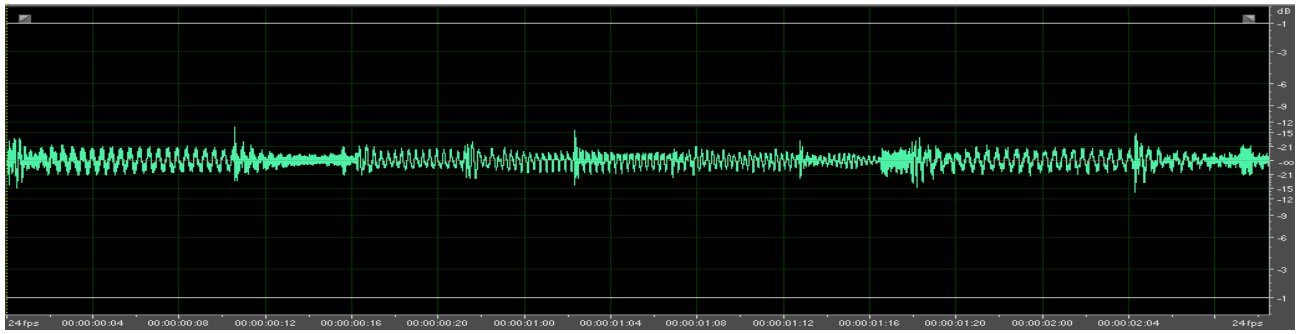


- Bossa.wav - normalized

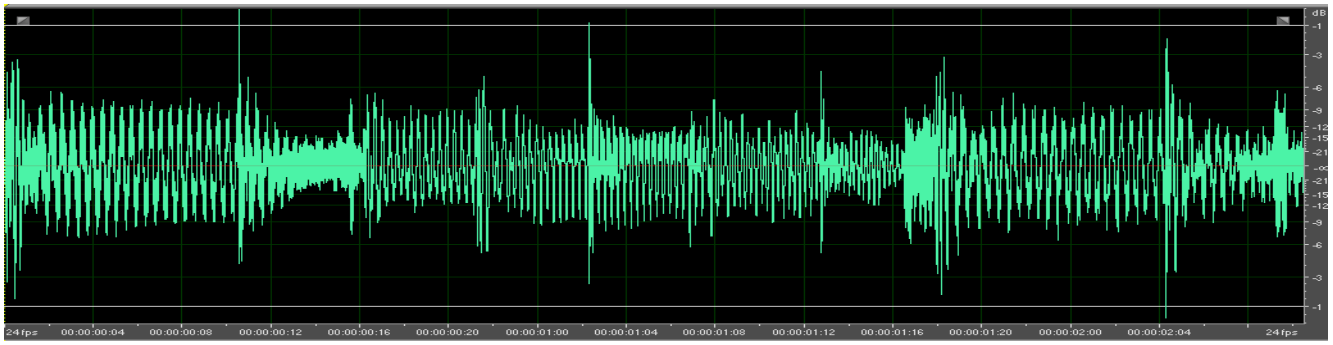


Dynamics Processing - Example

- Bossa.wav - compressed



- Bossa.wav – compressed + normalized

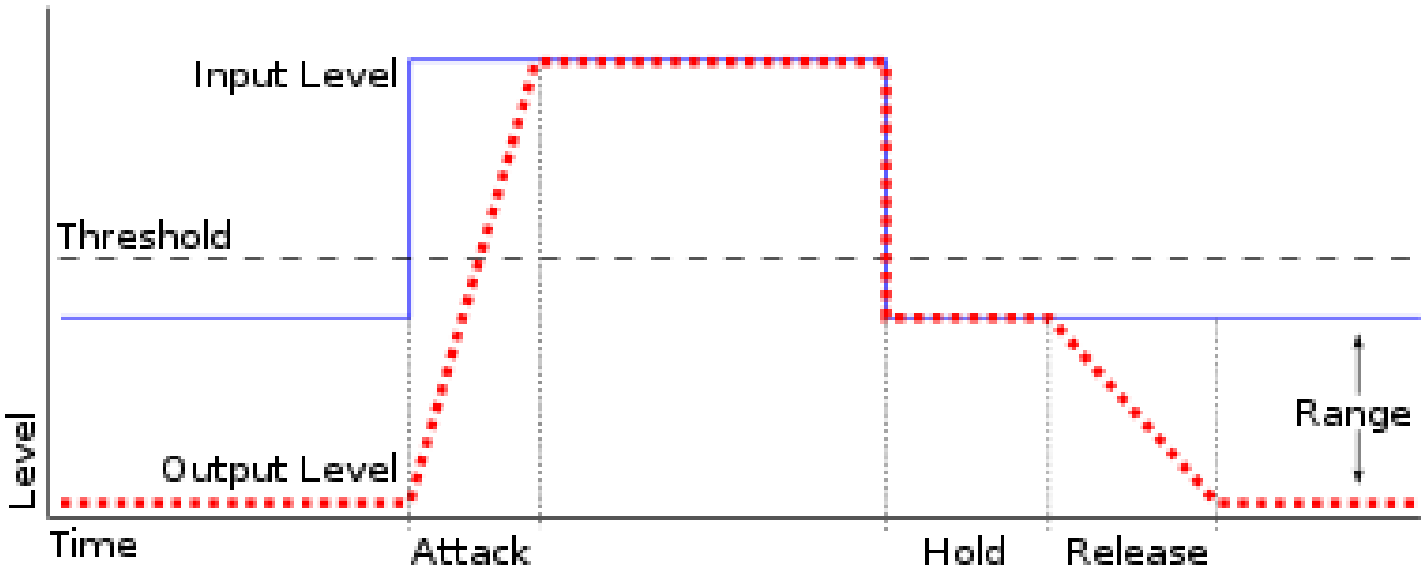


Audio Restoration

- In this section, we introduce two basic types of audio restoration to alleviate the background noise that arises from the microphone, air, disk …etc.
  - Noise gating
  - Noise reduction

Noise Gating

- A **noise gate** allows a signal to pass through only when it is above a set threshold.
- It is used when the level of the signal is above the level of the noise. **It does not remove noise from the signal**. When the gate is open, both the signal and the noise will pass through.



http://en.wikipedia.org/wiki/Noise_gate

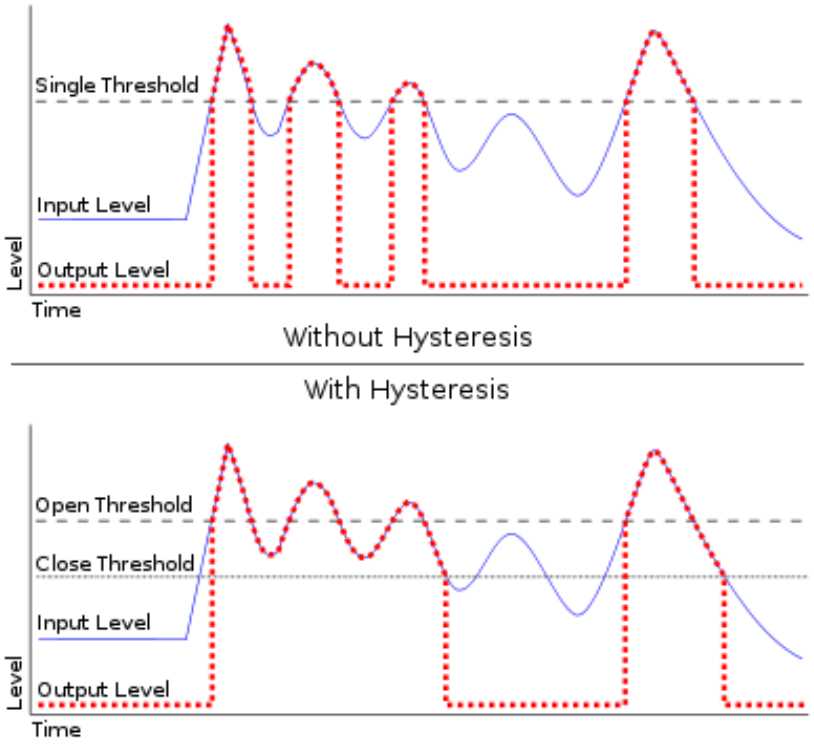
Noise Gating

- **Reduction Level**: the amplitude to which you want the below-threshold samples to be reduced.
- **Attack**: the attack time indicates how quickly you want the gate to open when the signal goes above the threshold, like fade-in.
- **Hold**: the amount of time the gate will stay open after the signal falls below the threshold.
- **Release**: The release time indicates how quickly you want the gate to close, like fade-out.

Noise Gating

- If the signal keeps moving back and forth around the threshold, the gate will open and close continuously, creating a kind of **chatter.**

- The **hysteresis** control indicates the difference between the value **n** that caused the gate to open and the value **m** that will cause it to close again. If *n* – *m* is large enough to contain the fluctuating signal, the noise gate won't cause chatter.

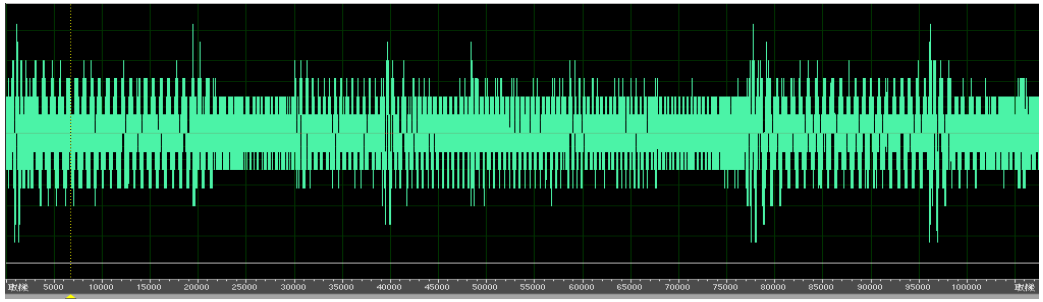


Noise Reduction

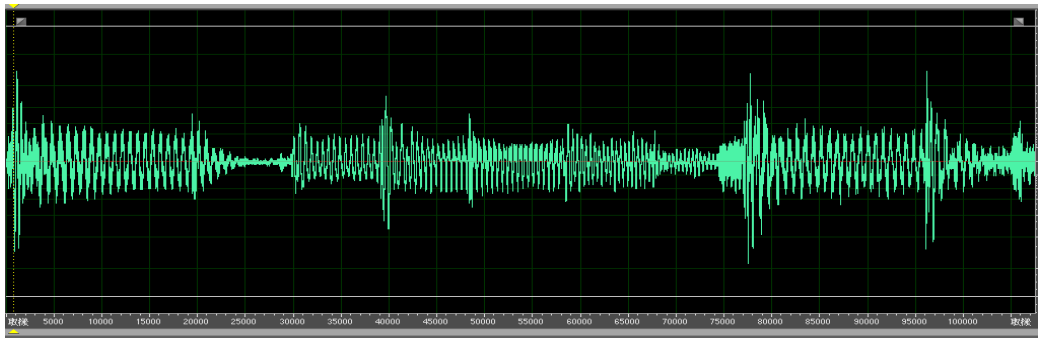
- Steps for noise reduction:
  1. Get a profile of the background noise. This can be done by selecting an area that should be silent, but that contains a hum or buzz.
  2. Determine the frequencies in the noise and their corresponding amplitude levels.
  3. The entire signal is processed in sections (FFT). The frequencies in each section are analyzed and compared to the profile, and if these sections contain frequency components similar to the noise, they can be eliminated below certain amplitudes.

Noise Reduction - Example

- Before noise reduction



- After noise reduction



Digital Audio Filter

- A digital audio filter is a linear system that changes the amplitude or phase of one or more frequency components of an audio signal.

- Types of digital audio filter
  - **FIR (finite-impulse response) filter**
  - **IIR (infinite-impulse response) filter**

FIR Filter

- Let $x(n)$ be an audio signal of N samples for $0 \leq n \leq N-1$

  $y(n)$ be the filtered signal

  $h(n)$ be the convolution mask

- The FIR filter function is defined by

$$y(n) = h(n) \otimes x(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

$$where\ x(n-k) = 0\ if\ n-k < 0$$

- The number of the coefficients of h is the **order of the filter.**

IIR Filter

- The infinite form of the FIR filter function is defined by

$$y(n) = h(n) \otimes x(n) = \sum_{k=0}^{\infty} h(k)x(n-k)$$

$$where\ x(n-k) = 0\ if\ n-k < 0$$

- However, finding the values $h(n)$ for an infinitely long mask is impossible. The equation can be transformed to a more manageable difference equation form.

IIR Filter

- The recursive form of IIR filter is

$$y(n) = h(n) \otimes x(n) = \sum_{k=0}^{N-1} a_k x(n-k) - \sum_{k=1}^{M} b_k y(n-k)$$

a_k is the coefficient of the forward filter

b_k is the coefficient of the feedback filter

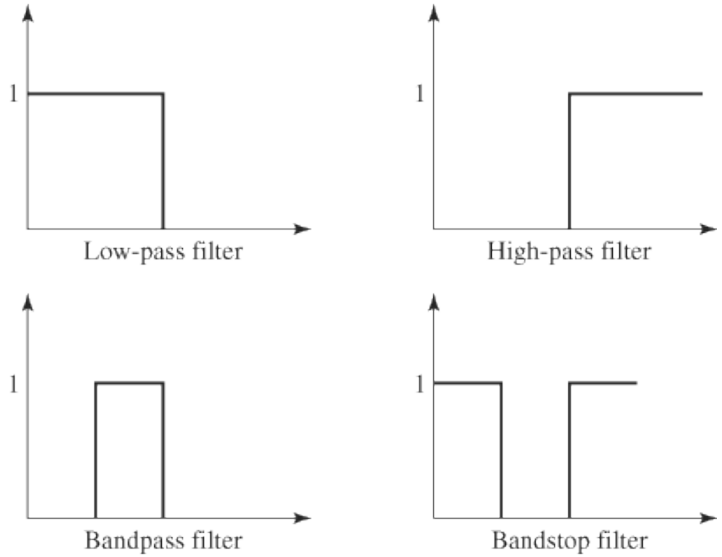
- y(n) depends on present and past input samples as well as on past outputs.

Impulse and Frequency Response

- The convolution mask **h(n)** for an FIR or IIR filter is sometimes referred to as the **impulse response**.
- The **frequency response**, **H(z)**, represents **h(n)** in frequency domain.
- A **frequency response graph** describes how a filter acts on an audio signal.

Filters in Audio Processing

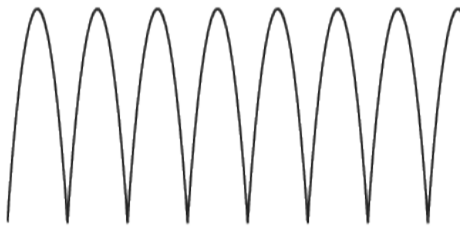
- **Band filters**
  - **low-pass filter**—retains only frequencies below a given level.
  - **high-pass filter**—retains only frequencies above a given level.
  - **bandpass filter**—retains only frequencies within a given band.
  - **bandstop filter**—eliminates all frequencies within a given band.



Horizontal axis: Frequency component
Vertical axis: Fraction of frequency component **retained** in filtered signal

Filters in Audio Processing

- **Comb filter —** add delayed versions of a wave to itself, resulting in phase cancellations that can be perceived as echo. Phase cancellations eliminate frequency components when two sine waves that are out-of-phase with each other are summed. Thus, the frequency response has the shape of a comb.



Frequency response of a comb filter

One-fold echo: $\mathbf{a_k}$ = [1, 0, 0, 0, ..., 0, 0.8] , $\mathbf{b_k}$ = [1]
(That is, 3199 zeros between 1 and 0.8.)
The output of the filter is: **y[n] = x[n] + 0.8*x[n-3200]**

Multiple-fold echo: $\mathbf{a_k}$ = [1], $\mathbf{b_k}$ = [1, 0, 0, 0, ..., 0, -0.8]
(That is, 3199 zeros between 1 and -0.8.)
The output of the filter is: **y[n] = x[n] + 0.8*y[n-3200]**

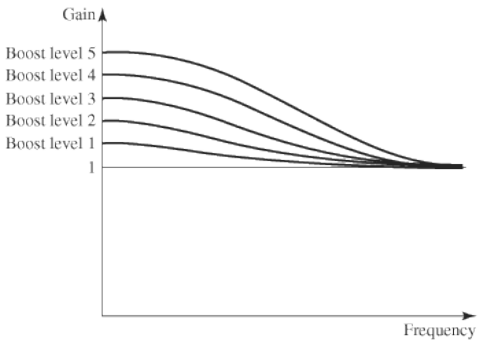
Filters in Audio Processing

- **Shelving filters —** shelving filters are similar to low- and high-pass filters except that they boost or cut frequencies up to a certain frequency.

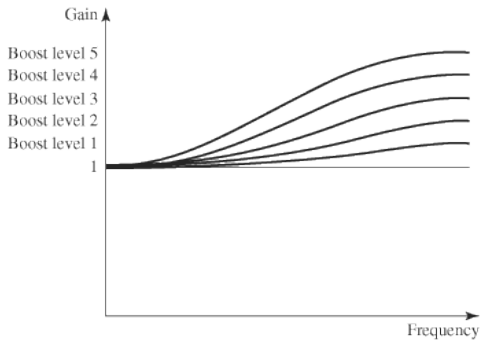
Low-shelf

High-shelf

Boosting

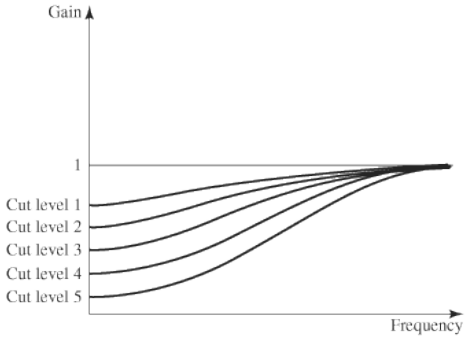


(a) Low-shelf filter for boosting low frequencies

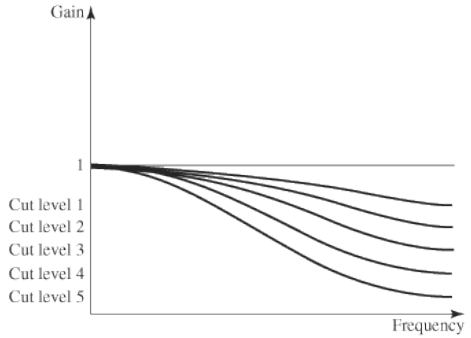


(b) High-shelf filter for boosting high frequencies

Cutting



(c) Low-shelf filter for cutting low frequencies

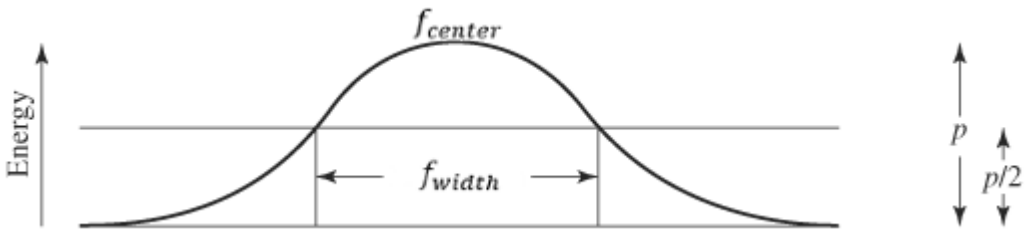


(d) High-shelf filter for cutting high frequencies

Filters in Audio Processing

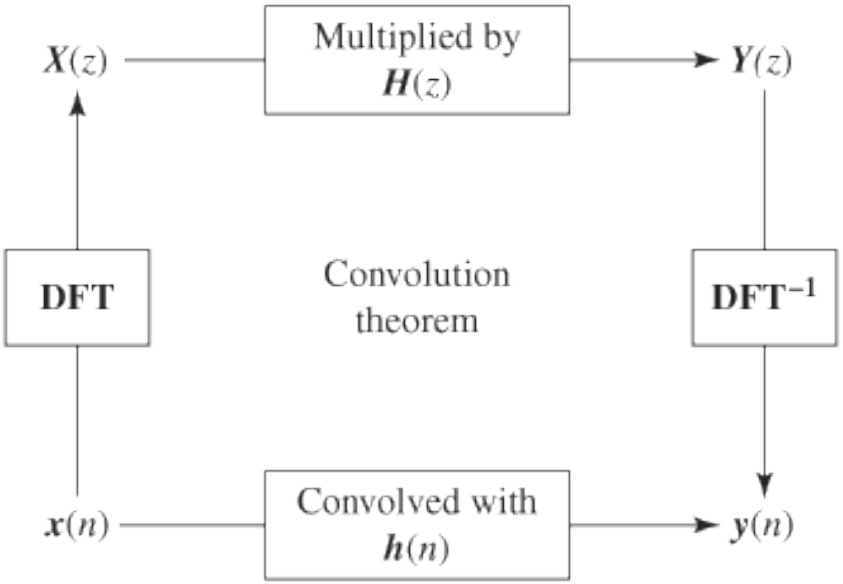
- **Peaking filter —** Ideally, for a bandpass filter, the unwanted frequencies would be filtered out entirely, but in reality this is not possible.
- The frequency response looks more like the bell curve. This type of filter is sometimes called a **peaking filter**.
- Given the graph of a peaking filter, let f_{width} be the width of the peak measured at the points ½ times the peak's height, and let f_{center} be the frequency at the geometric center of the peak, both in Hz. Then the Q-factor, Q, is defined as

$$Q = \frac{f_{center}}{f_{width}}$$



Relationship Between Convolution and Fourier Transform

- Let ***H***(*z*) be the discrete Fourier transform of a convolution filter ***h***(*n*), and let ***X***(*z*) be the discrete Fourier transform of a digital audio signal ***x***(*n*). Then $y(n) = h(n) \otimes x(n)$ is equivalent to the inverse discrete Fourier transform of ***Y***(*z*), where ***Y***(*z*) = ***H***(*z*)***X***(*z*).



Equivalent operations in time and frequency domains

FIR Filter Design – Terms

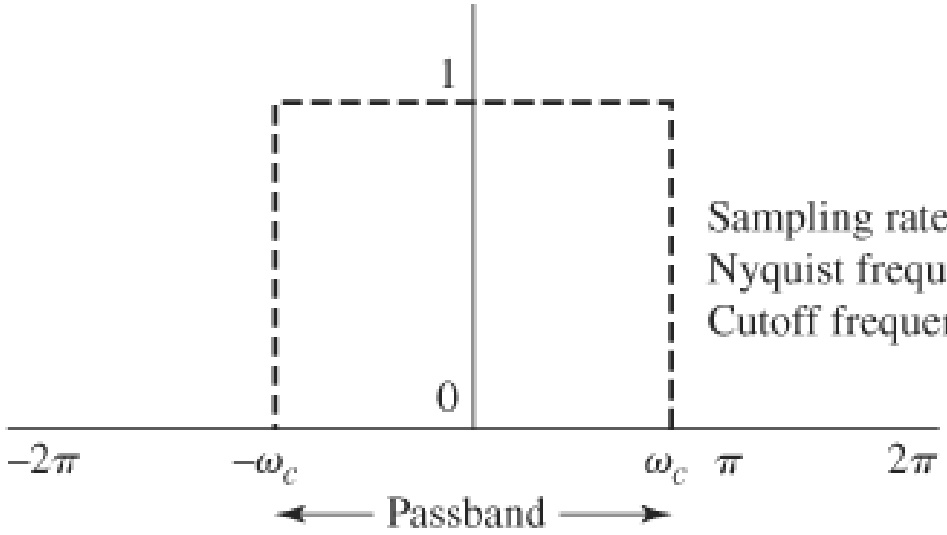
- The convolution mask **h**(n) is also called the **impulse response**, representing a filter in the **time domain**.

- Its counterpart in the frequency domain, the **frequency response H**(z), is also sometimes referred to as the **transfer function**.

- A frequency response graph can be used to show the desired frequency response of a filter you are designing.

FIR Filter Design – Ideal Filter

- An ideal low-pass frequency response graph, normalized.
- In the graph, angular frequency is on the horizontal axis. The vertical axis represents the fraction of each frequency component to be permitted in the filtered signal.
- The cutoff frequency ω_c must be less than π.(Nyquist theorem)

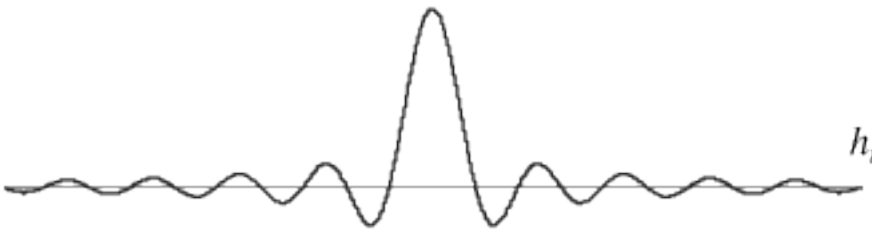


$$\omega_c = 2\pi \frac{f_c}{f_{sample}}$$

Figure 5.42

FIR Filter Design – Ideal Filter

- The frequency response graph shown in Figure 5.42 is an example of a **rectangle function.**
- If it is an idealized form of ***H***(*z*), then what would be the corresponding ideal impulse response, an idealized ***h***(*n*)?
- The inverse Fourier transform of a rectangle function in the frequency domain is a **sinc function** in the time domain.



$$h_{ideal}(n) = \frac{\sin(2\pi f_c n)}{\pi n} \text{ for } -\infty \le n \le \infty, n \ne 0 \text{ and } 2f_c \text{ for } n = 0$$

FIR Filter Design – Ideal Filter

TABLE 5.2 Equations for Ideal Impulse Responses for Standard Filters, Based on Cutoff Frequency f_c and Band Edge Frequencies f_1 and f_2

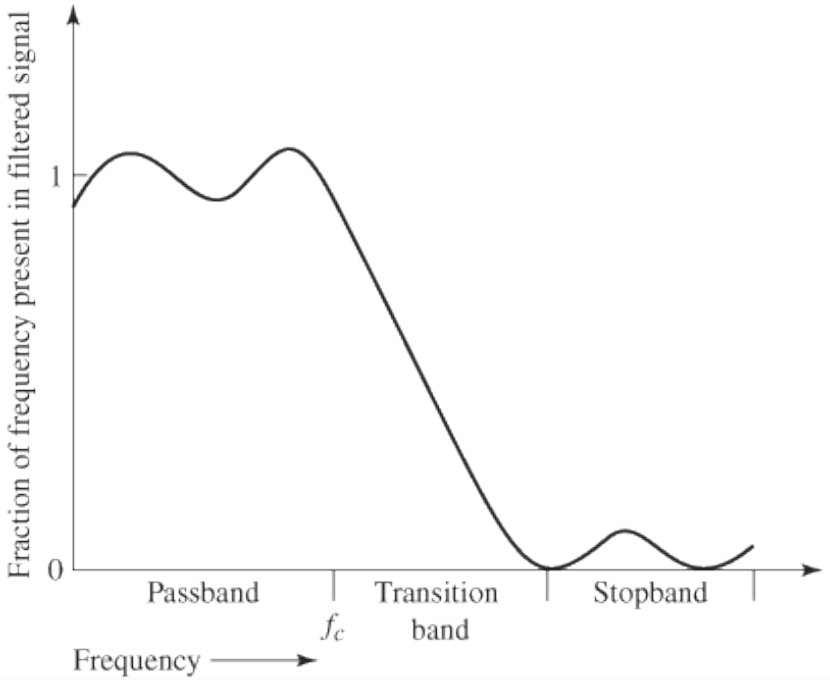
Type of filter	$h_{ideal}(n)$, $n \neq 0$	$h_{ideal}(0)$
Low-pass	$\frac{\sin(2\pi f_c n)}{\pi n}$	$2f_c$
High-pass	$-\frac{\sin(2\pi f_c n)}{\pi n}$	$1 - 2f_c$
Bandpass	$\frac{\sin(2\pi f_2 n)}{\pi n} - \frac{\sin(2\pi f_1 n)}{\pi n}$	$2(f_2 - f_1)$
Bandstop	$\frac{\sin(2\pi f_1 n)}{\pi n} - \frac{\sin(2\pi f_2 n)}{\pi n}$	$1 - 2(f_2 - f_1)$

FIR Filter Design – Ideal to Real

- A sinc function goes on infinitely in the positive and negative directions … how can we implement an FIR filter?
- One way is to multiply the ideal impulse response by a **windowing function**. The purpose of the windowing function is to make the impulse response finite.
- However, making the impulse response finite results in a frequency response that is less ideal, containing "ripples".

FIR Filter Design - Realistic

- The **passband** corresponds to the frequencies the filter tries to retain. The **stopband** corresponds to the frequencies the filter attenuates or filters out.
- In a real filter, a **transition band** lies between passband and stopband, and the slope is not infinitely steep, as in an ideal filter.



Frequency response of a realistic low-pass filter

FIR Filter Design – Windowing Function

- Rectangular windowing function
  - 1
- Hanning windowing function
  - $w(n) = 0.5 + 0.5\cos(\frac{2\pi n}{N})$
- Hamming windowing function
  - $w(n) = 0.54 + 0.46\cos(\frac{2\pi n}{N})$
- Blackmann windowing function
  - $w(n) = 0.42 + 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos(\frac{4\pi n}{N-1})$

FIR Filter Design - Algorithm

```
/*Input: f_c, the cutoff frequency for the lowpass filter, in Hz
         f_samp, the sampling frequency of the audio signal to be filtered, in Hz
         N, the order of the filter; assume N is odd
  Output: a low-pass FIR filter in the form of an N-element array */

/*Normalize f_c and ω_c so that π is equal to the Nyquist angular frequency*/
f_c = f_c/f_samp
ω_c = 2*π*f_c
middle = N/2 /*Integer division, dropping remainder*/

/*Create the filter using the low-pass filter function from Table 5.2*/
/*Put a dummy value in for n = 0 to avoid a divide by 0 error)*/
for n = −N/2 to N/2
    if (n = 0) fltr(middle) = 1
    else fltr(n + middle) = sin(2*π*f_c*n)/(π*n)
fltr(middle) = 2*f_c

/*Multiply the elements of fltr by a windowing function chosen from Table 5.3. We use the Hanning window
function */
for n = 0 to N-1
    fltr(n) = fltr(n) * (0.5 + 0.5*cos((2*π*n)/N))
```

Digital Audio Compression

- Time-Based Compression Methods
  - No need to transform the data into frequency domain.
  - A-law encoding, μ-law encoding, etc.
  - These methods are often considered conversion techniques rather than compression methods.
- The most effective audio compression methods require some information about the frequency spectrum of the audio signal. These compression methods are based on **psychoacoustical modeling** and **perceptual encoding**.

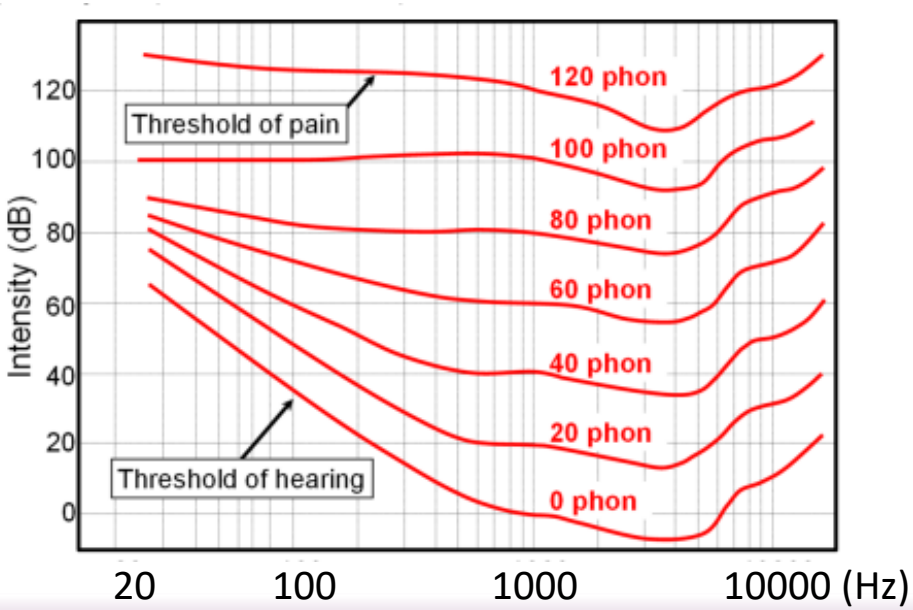
Psychoacoustics

- Psychoacoustics (心理聲學)
  - The study of subjective human perception of sounds.
  - The study of all the psychological interactions between humans and the world of sound.
- Psychoacoustical tests have shown that there is a great deal of nonlinearity in human sound perception.
- The octave from middle C (called C4) to C5 ranges from 261.63 Hz to 523.25 Hz, while C5 to C6 ranges from 523.25 to 1046.50 Hz. But the distance from C4 to C5 subjectively sounds the same as the distance from C5 to C6.

Psychoacoustics

- Humans hear best in the 1000 to 5000 Hz range, the frequency range of human speech.
- For a 100Hz and 1000Hz tone, the 100Hz tone needs larger amplitude to sound equally loud as the 1000Hz.
- Similarly, for a 10000Hz and 1000Hz tone, the 10000Hz tone needs larger amplitude to sound equally loud as the 1000Hz.

- The **phon** is a unit of perceived loudness level varying from frequencies
- 1 phon = 1 dB SPL at 1000 Hz
- 0 phon is the limit of perception, and inaudible sounds have negative phon levels

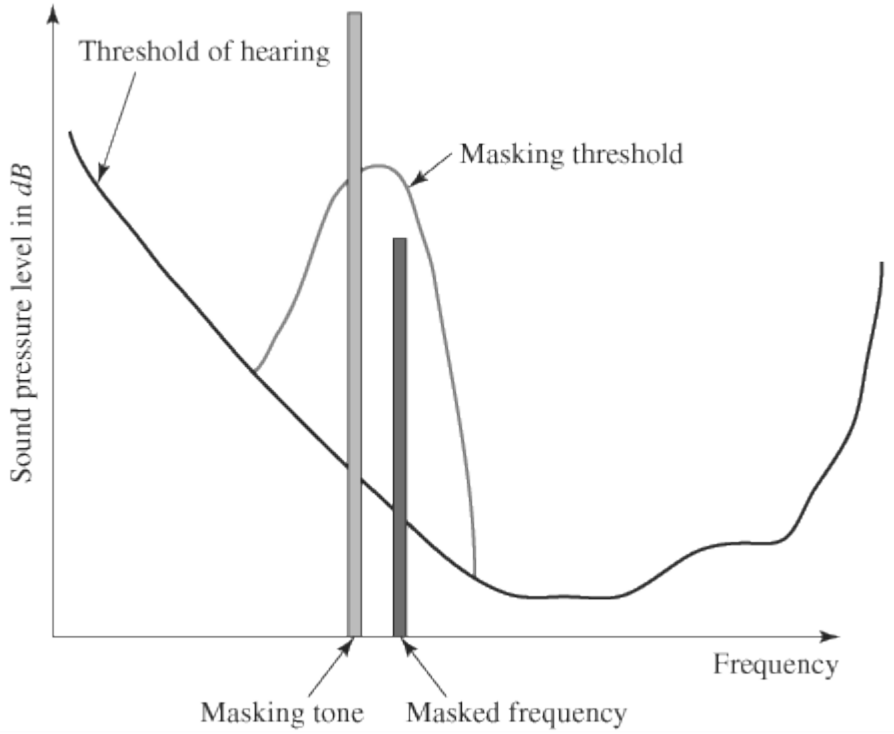


Psychoacoustics

- At low frequencies, we can distinguish between sound waves that are only a few Hz apart. At high frequencies, the frequencies must be a hundred or more Hz apart for us to hear the difference. The reason is that the ear is divided into frequency bands called ***critical bands***.
- Human ear perceive different frequencies by the response of different critical bands.
- Critical bands are narrower for low-frequency than for high-frequency sounds. Between 1 and 500 Hz, bands are about 100 Hz in width. The critical band at the highest audible frequency is over 4000 Hz wide.

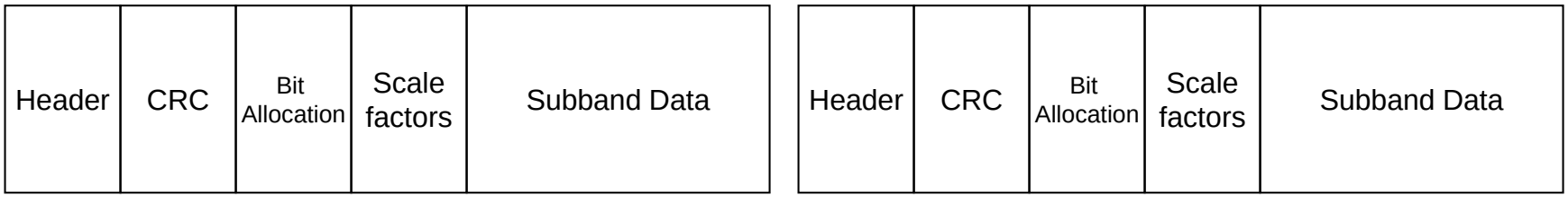
Psychoacoustics

- **Masking** - Within a small window of time, the loudest frequency sound can overpower the others in the critical band, making human unable to hear the other frequencies.



A Sketch of Compression

- A small window of time called a ***frame*** is moved across a sound file.
- In each frame, use different filters to divide the frame into bands of different frequencies.
- Calculate the masking curve for each band.
- Requantize the samples using fewer bits such that the quantization error is below the masking curve. That is, make sure the noise is inaudible.



Frame 1

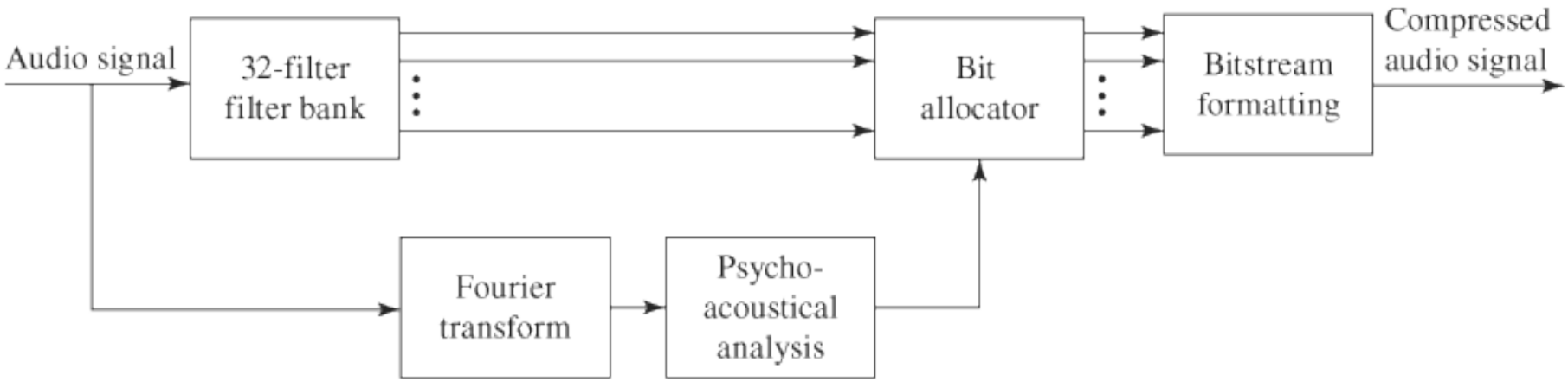
Frame 2

Overview of MPEG

- Acronym for the ***Motion Picture Experts Group,*** a family of compression algorithms for both digital audio and video.
- MPEG has been developed in a number of phases: MPEG-1, MPEG-2, MPEG-4, MPEG-7.
- MPEG-1 covers CD-quality audio suitable for video games.
- MPEG audio is also divided into three layers: Audio Layers I, II, and III.
- Each higher layer is more computationally complex, and generally more efficient at lower bitrates than the previous.
- The well-known MP3 audio file format is actually MPEG-1 Audio Layer III.

MPEG-1 Audio Compression

- Basic concepts of MPEG-1 compression



MPEG-1 Audio Compression - Step 1

- Divide the audio file into frames and analyze the psychoacoustical properties of each frame individually.
  - To analyze the masking phenomenon, we must look at a small piece of time because it happens when different frequencies are played at close to the same time.
  - Frames can contain 384, 576, or 1152 samples, depending on the MPEG phase and layer.
  - For the remainder of these steps, it is assumed that we're operating on an individual frame.

MPEG-1 Audio Compression - Step 2

- By applying a bank of filters, separate the signal into frequency bands.
  - The samples are divided into frequency bands for psychoacoustical analysis. Each filter removes all frequencies except for those in its designated band.
  - The use of filter banks is called **subband coding.** In MPEG-1, the number of filters is 32.
  - In MPEG-1 Layers I and II, the frequency bands created by the filter banks are uniform in size, which doesn't match the width of the critical bands in human hearing. Layer III models critical bands more closely than layer I and II.

MPEG-1 Audio Compression - Step 3

- Perform a Fourier transform on the samples in each band in order to analyze the band's frequency spectrum.
  - After the Fourier transform, we can know exactly how much of each frequency component occurs in each band.
  - From the frequency spectrum, a masking curve can be produced for each band.

MPEG-1 Audio Compression - Step 4

- Analyze the influence of tonal and nontonal elements in each band. (Tonal elements are simple sinusoidal components, such as frequencies related to melodic and harmonic music. Nontonal elements are transients like the strike of a drum or the clapping of hands.)
  - The longer the time window is in frequency analysis, the better the frequency resolution, but the worse the time resolution. This in turn can mean that transient signals may not be properly identified. Therefore we'd better identify the transient first.

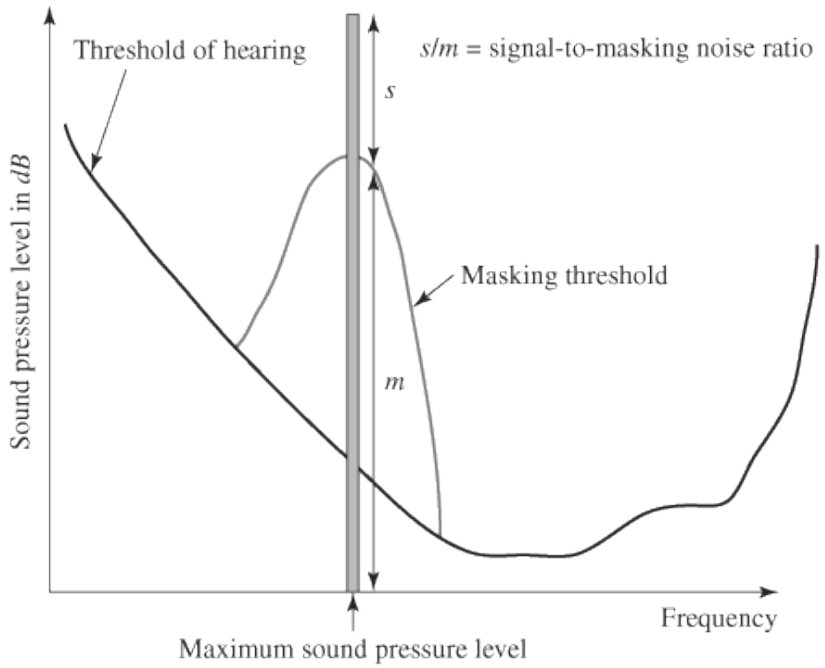
MPEG-1 Audio Compression - Step 5

- Determine how much each band's influence is likely to spread to neighboring frequency bands.
  - It isn't sufficient to deal with bands entirely in isolation from each other, since there can be a masking effect between bands.

MPEG-1 Audio Compression - Step 6

- Find the masking threshold and *signal-to-mask ratio* (*SMR*) for each band, and determine the bit depth of each band accordingly.

- SMR: the ratio between the peak sound pressure level and the masking threshold.



MPEG-1 Audio Compression - Step 6

- The masking phenomenon causes the noise floor within a band to be raised. When the noise floor is high relative to the maximum sound pressure level within a band, then fewer bits are needed.
- Fewer bits create more quantization noise, but it doesn't matter if that quantization noise is below the masking threshold. The noise won't be heard anyway.

MPEG-1 Audio Compression - Step 7

- Quantize the samples for the band with the appropriate number of bits, possibly following this with Huffman encoding.
  - MPEG-1 Layers 1 and 2 use linear quantization, while MP3 uses nonlinear.

MPEG-1 Audio Compression

ALGORITHM 5.2

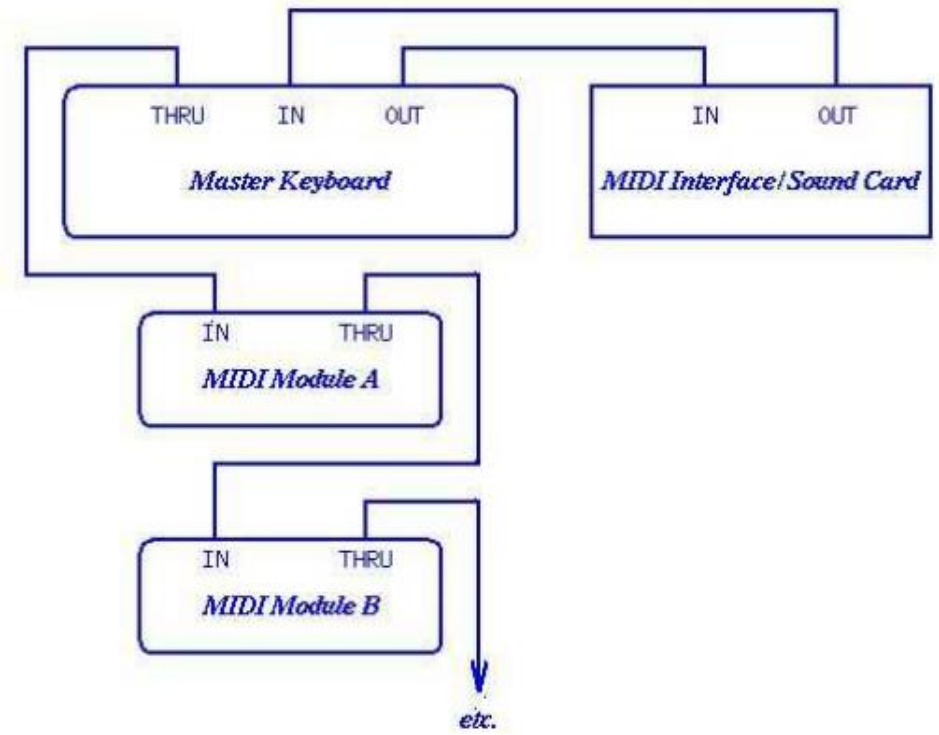
```
algorithm MPEG-1_audio
/*Input: An audio file in the time domain
  Output: The same audio file, compressed*/
{
  Divide the audio file into frames
  For each frame {
    By applying a bank of filters, separate the signal into frequency bands.
    For each frequency band {
      Perform a Fourier transform to analyze the band's frequency spectrum
      Analyze the influence of tonal and nontonal elements (i.e., transients)
      Analyze how much the frequency band is influenced by neighboring bands
      Find the masking threshold and signal-to-mask ratio (SMR) for the band,
      and determine the bit depth in the band accordingly
      Quantize the samples from the band using the determined bit depth
      Apply Huffman encoding (optional)
    }
    Create a frame with a header and encoded samples from all bands
  }
}
```

MIDI

- Musical Instrument Digital Interface (MIDI)
- No Longer Exclusively the Domain of Musicians.
- MIDI provides a very low bandwidth alternative on the Web:
  - transmit musical
  - certain sound effects data
- Now used as a compression control language
  - MPEG-4
  - HTML5

Definition of MIDI

- A protocol that enables computers, synthesizers, keyboards and other musical devices to communicate with each other.



MIDI System

Synthesizer/Sampler

- It is a sound generator (various pitch, loudness, tone color)
- Can use a variety of synthesis or Sample-based synthesis to make sound.
- A good (musician's) synthesizer often has a microprocessor, keyboard, control panels, memory, etc.
- For our purposes we define a synthesizer as the tone generation unit.


MIDI Messages

MIDI messages are used by MIDI devices to communicate with each other.

MIDI messages are very low bandwidth:

- Note On Command
  - Which Key is pressed
  - Which MIDI Channel (what sound to play)
  - 3 Hexadecimal Numbers
- Note Off Command Similar
- Other command (program change) configure sounds to be played.

MIDI Command Example

MIDI Note On Example:

A Note On message is followed by two bytes, one to identify the note, and one to specify the velocity.

To play:

- Note number **80** (Hex **50**)
- With maximum velocity **127** (Hex **7F**)
- On channel **13** (Hex **C**),

The MIDI device would send these three hexadecimal byte values:

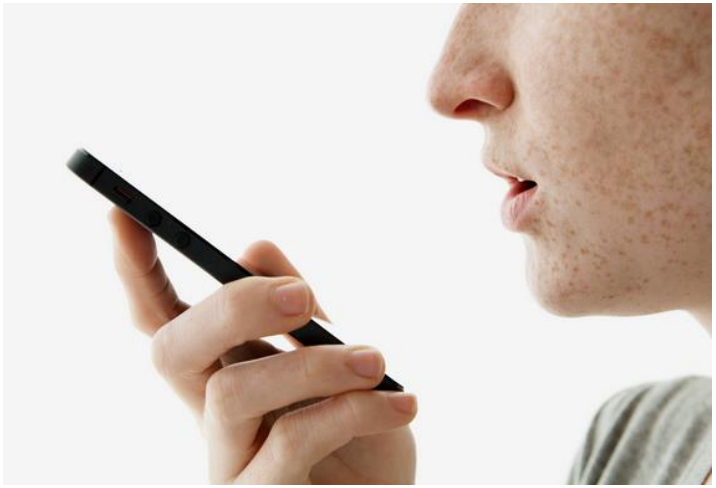
9C 50 7F

MPEG-4 Structured Audio

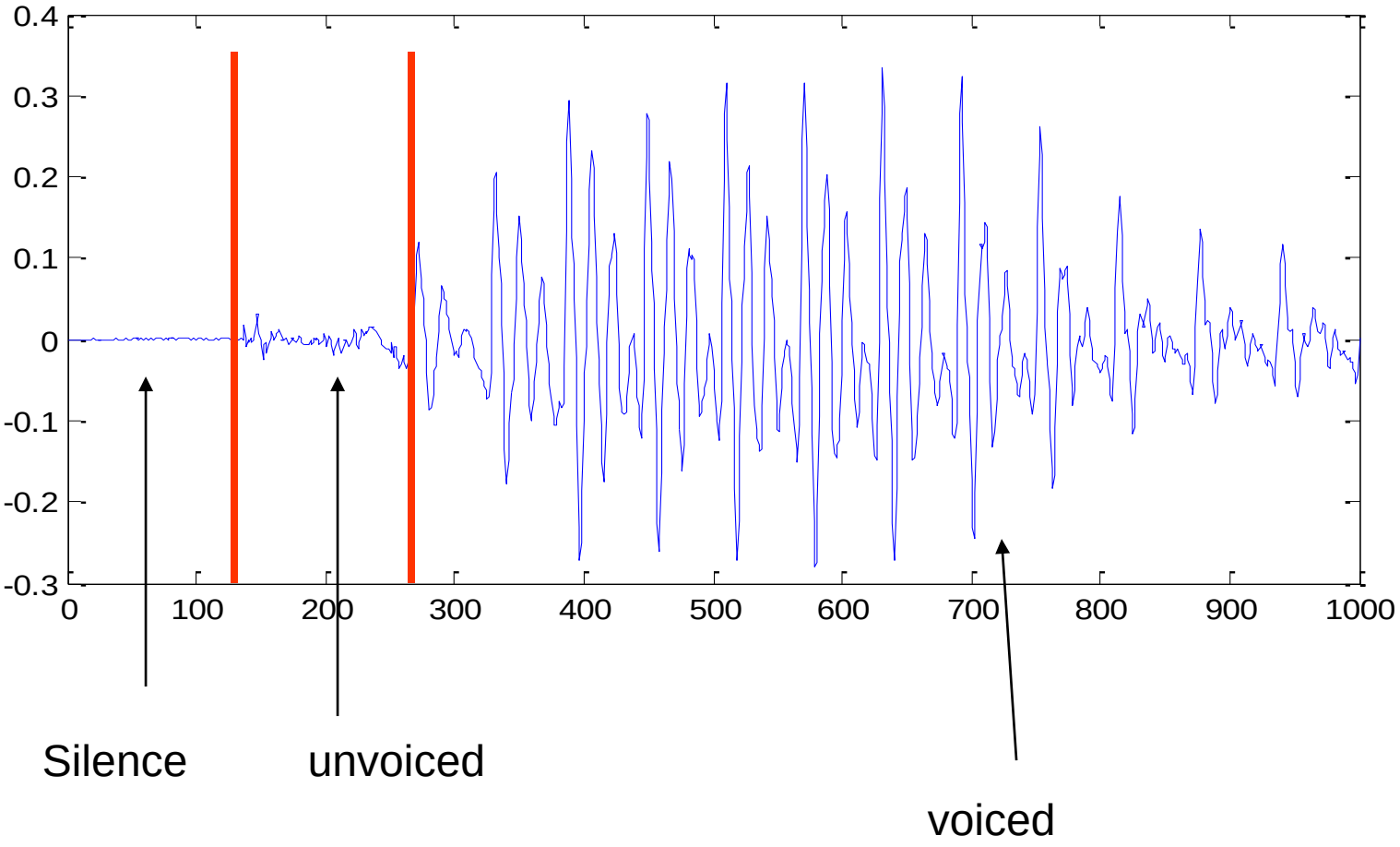
- Basic ideas of compression via bit quantization studied shortly: used as integral part of audio format — MP3, real audio etc.
- MPEG-4 audio — actually combines compression synthesis and MIDI to have a massive impact on compression.
- Basic Idea: **MIDI** + **Synthesis** encode what note to play and how to play it with a small number of parameters
  — Much greater reduction than simply having some encoded bits of audio.

Speech Processing

- Speech is our basic communication tool.
- We have been hoping to be able to communicate with machines using speech.
- Speech recognition & synthesis

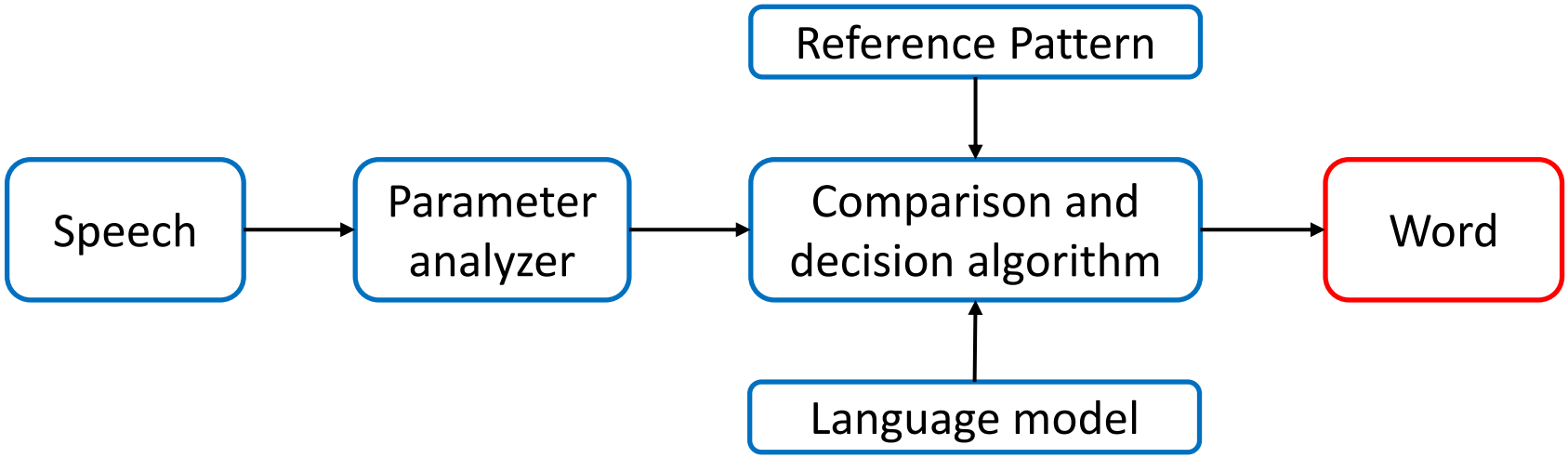


Voiced and Unvoiced Speech



Speech Recognition

- Speech recognition is the foundation of human computer interaction using speech.
- Speech recognition in different contexts
  - Dependent or independent on the speaker.
  - Discrete words or continuous speech.
  - Small vocabulary or large vocabulary.
  - In quiet environment or noisy environment.



Speech Synthesis

- Speech synthesis is to generate (arbitrary) speech with desired properties (pitch, speed, loudness, articulation mode, etc.)
- Speech synthesis has been widely used for text-to-speech systems and different telephone services.
- The easiest and most often used speech synthesis method is waveform concatenation.

Increase the pitch without changing the speed

Music Recommendation

- Automatic music recommendation has become an increasingly relevant problem in recent years, since a lot of music is now sold and consumed digitally.
  - Online music stores and streaming services such as iTunes, Spotify, Google Play, and KKBOX.
  - Automatic music recommendation allows listeners to discover new music that matches their tastes, and enables online music stores to target their wares to the right audience.

Music Recommendation

- Current music recommendation techniques
  - Collaborative filtering
  - Content-based methods
  - Context-based methods
  - Hybrid methods

Music Recommendation

- Collaborative filtering
  - Collaborative filtering recommends songs by considering the preferences of other like-minded users.
  - For instance, if user A and B have similar music preferences, then songs liked by A but not yet considered by B will be recommended to B.
  - One of the biggest problem is that **new and unpopular songs cannot be recommended**: if there is no usage data to analyze. This is the so-called **cold-start problem**.

Music Recommendation

- Content-based methods
  - Content-based methods recommend songs that have similar audio content to the user's preferred songs.
  - This kind of approach requires the definition of a suitable similarity metric.

Music Recommendation

- Context-based methods
  - Context-based methods recommend songs to match various aspects of the user context (e.g., activities, environment, or physiological states. They have become increasingly popular in recent years with the advent of sensor-rich and computationally powerful smartphones.

Music Recommendation

- Hybrid methods
  - Hybrid methods combine two or more of the above methods.

Summary

- Audio signal sampling, digitization
- Audio dithering
- Noise shaping
- Frequency analysis
- Dynamic processing
- Audio restoration
- Digital audio filters
- FIR filter design
- Perceptual encoding
- MPEG-1 compression
- MIDI
- Speech Processing
- Music Recommendation