

# Sound Analysis

Tamara Berg  
Advanced Multimedia

# Reminder

HW1 – due Feb 21

Questions?

---

Thurs Feb 23

Matlab sound lab

Bring headphones!!!

---

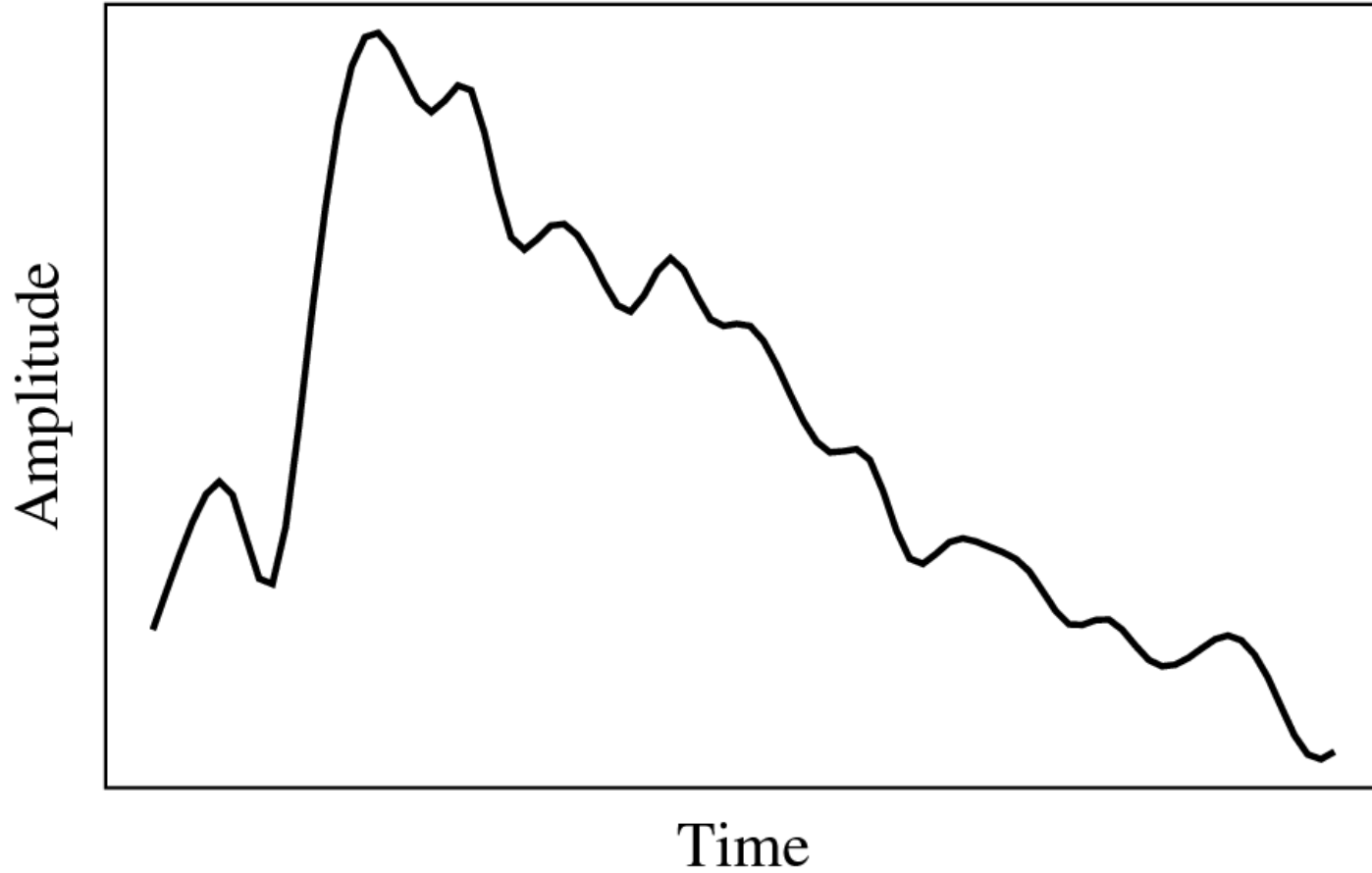
# Summary

Sound in a computer is represented by a vector of values – discrete samples of the continuous wave.

[x1 x2 x3 x4 x5 x6....]

Here you have two things to decide:

# Sound Wave



# Summary

Sound in a computer is represented by a vector of values – discrete samples of the continuous wave.

[x1 x2 x3 x4 x5 x6....]

Here you have two things to decide:

How often to sample the signal (sampling rate).

How to quantize your continuous sound signal into a set of discrete values.

# Audio Filtering

- Prior to sampling and AD conversion, the audio signal is also usually *filtered* to remove unwanted frequencies. The frequencies kept depend on the application:
  - (a) For speech, typically from 50Hz to 10kHz is retained, and other frequencies are blocked by the use of a **band-pass filter** that screens out lower and higher frequencies.
  - (b) An audio music signal will typically contain from about 20Hz up to 20kHz.

# Audio Filtering

- Prior to sampling and AD conversion, the audio signal is also usually *filtered* to remove unwanted frequencies. The frequencies kept depend on the application:
  - (a) For speech, typically from 50Hz to 10kHz is retained, and other frequencies are blocked by the use of a **band-pass filter** that screens out lower and higher frequencies.
  - (b) An audio music signal will typically contain from about 20Hz up to 20kHz.
  - (c) At the DA converter end, high frequencies may reappear in the output — because of sampling and then quantization.
  - (d) So at the decoder side, a **lowpass** filter is used after the DA circuit.

Next class we'll see how to create filters in matlab and apply them to audio signals!

# Audio Quality vs. Data Rate

- The uncompressed data rate increases as more bits are used for quantization. Stereo: double the bandwidth. to transmit a digital audio signal.

Table 6.2: Data rate and bandwidth in sample audio applications

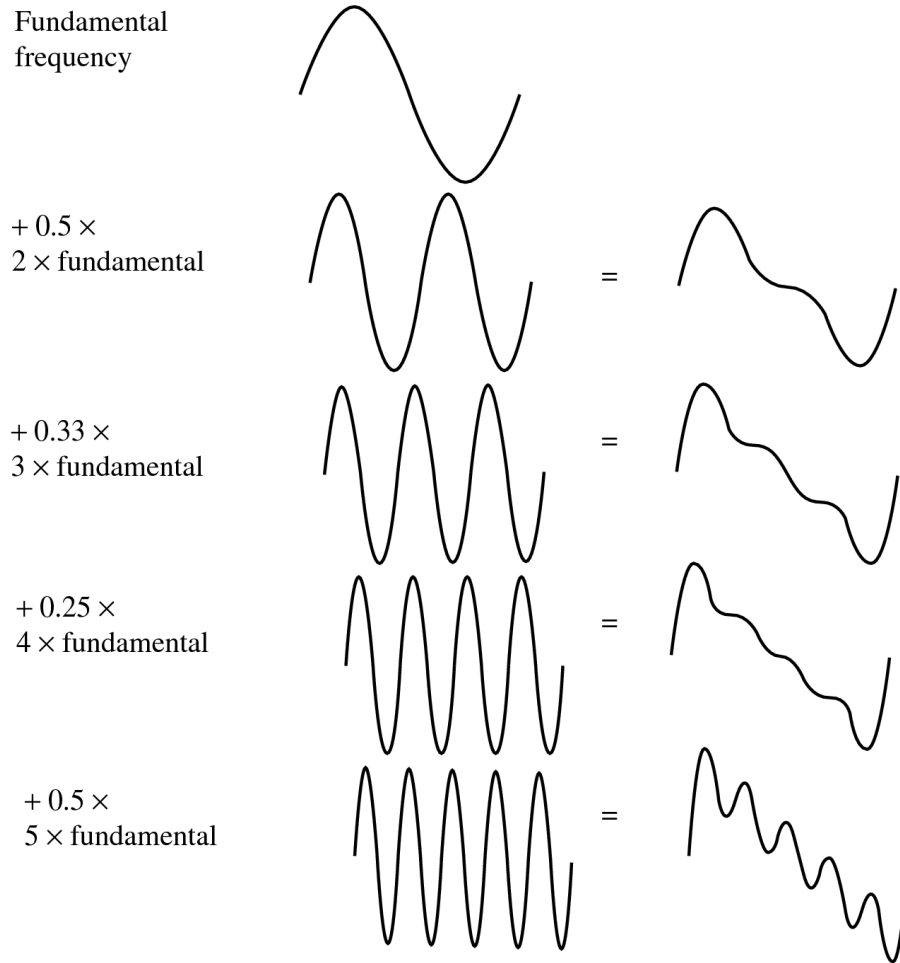
Quality	Sample Rate (Khz)	Bits per Sample	Mono / Stereo	Data Rate (uncompressed) (kB/sec)	Frequency Band (KHz)
Telephone	8	8	Mono	8	0.200-3.4
AM Radio	11.025	8	Mono	11.0	0.1-5.5
FM Radio	22.05	16	Stereo	88.2	0.02-11
CD	44.1	16	Stereo	176.4	0.005-20
DAT	48	16	Stereo	192.0	0.005-20
DVD Audio	192 (max)	24(max)	6 channels	1,200 (max)	0-96 (max)

# Creating Sounds

Two different approaches

FM

WaveTable



Signals can be decomposed into a weighted sum of sinusoids:  
Building up a complex signal by superposing sinusoids

# Synthetic Sounds

## 1. FM (Frequency Modulation): one approach to generating synthetic sound:

$$x(t) = A(t) \cos[\omega_c \pi t + I(t) \cos(\omega_m \pi t + \phi_m) + \phi_c]$$

$A(t)$  specifies overall loudness over time

$\omega_c$  specifies the carrier frequency

$\omega_m$  specifies the modulating frequency

$\phi_c$  Specify time-shifts for a more

$\phi_m$  interesting sound

$I(t)$  produces a feeling of harmonics (overtones) by changing the amount of the modulating frequency heard.

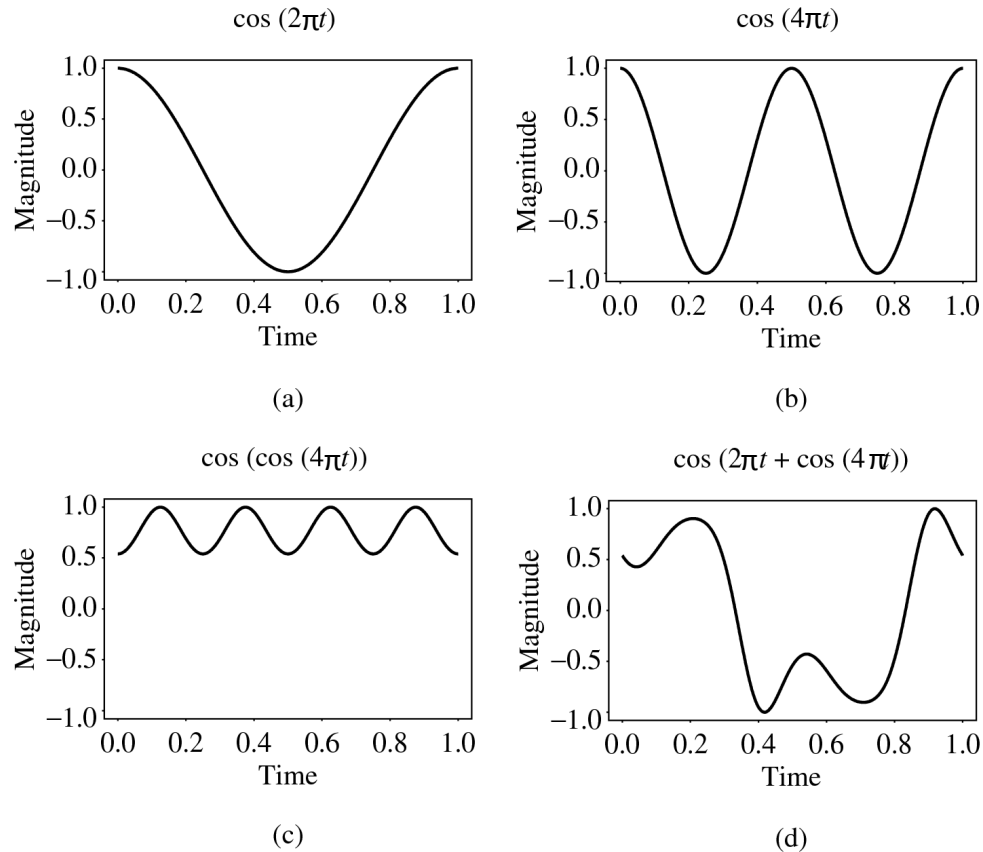


Fig. 6.7: Frequency Modulation. (a): A single frequency. (b): Twice the frequency. (c): Usually, FM is carried out using a sinusoid argument to a sinusoid. (d): A more complex form arises from a carrier frequency,  $2\pi t$  and a modulating frequency  $4\pi t$  cosine inside the sinusoid.

2. **Wave Table synthesis:** A more accurate way of generating sounds from digital signals. Also known, simply, as **sampling**.

In this technique, the actual digital samples of sounds from real instruments are stored. Since wave tables are stored in memory on the sound card, they can be manipulated by software so that sounds can be combined, edited, and enhanced.

## 6.2 MIDI: Musical Instrument Digital Interface

- **MIDI Overview**

- (a) MIDI is a protocol adopted by the electronic music industry in the early 80s for controlling devices, such as synthesizers and sound cards, that produce music and allowing them to communicate with each other.
- (b) MIDI is a scripting language — it codes “events” that stand for the production of sounds. E.g., a MIDI event might include values for the pitch of a single note, its duration, and its volume.

- (c) The MIDI standard is supported by most synthesizers, so sounds created on one synthesizer can be played and manipulated on another synthesizer and sound reasonably close.
  
- (d) Computers must have a special MIDI interface, but this is incorporated into most sound cards.
  
- (3) A MIDI file consists of a sequence of MIDI instructions (messages). So, would be quite small in comparison to a standard audio file.

# MIDI Concepts

- **MIDI channels** are used to separate messages.
  - (a) There are 16 channels numbered from 0 to 15. The channel forms the last 4 bits (the least significant bits) of the message.
  - (b) Usually a channel is associated with a particular instrument: e.g., channel 1 is the piano, channel 10 is the drums, etc.
  - (c) Nevertheless, one can switch instruments midstream, if desired, and associate another instrument with any channel.

- **Channel MIDI messages (which include a channel message)**
  - (a) For most instruments, a typical message might be a Note On message (meaning, e.g., a keypress and release), consisting of what channel, what pitch, and what “velocity” (i.e., volume).
  - (b) For percussion instruments, however, the pitch data means which kind of drum.

(c) The way a synthetic musical instrument responds to a MIDI message is usually by only **playing** messages that specify its channel.

- If several messages are for its channel (say play multiple notes on the piano), then the instrument responds to all of them, provided it is **multi-voice**, i.e., can play more than a single note at once (as opposed to violins).

- **System messages**

- Several other types of messages

- e.g. a general message for all instruments indicating a change in tuning or timing.

- A MIDI device can often also change the **envelope** describing how the amplitude of a sound changes over time.
- Fig. 6.9 shows a model of the response of a digital instrument to a Note On message:

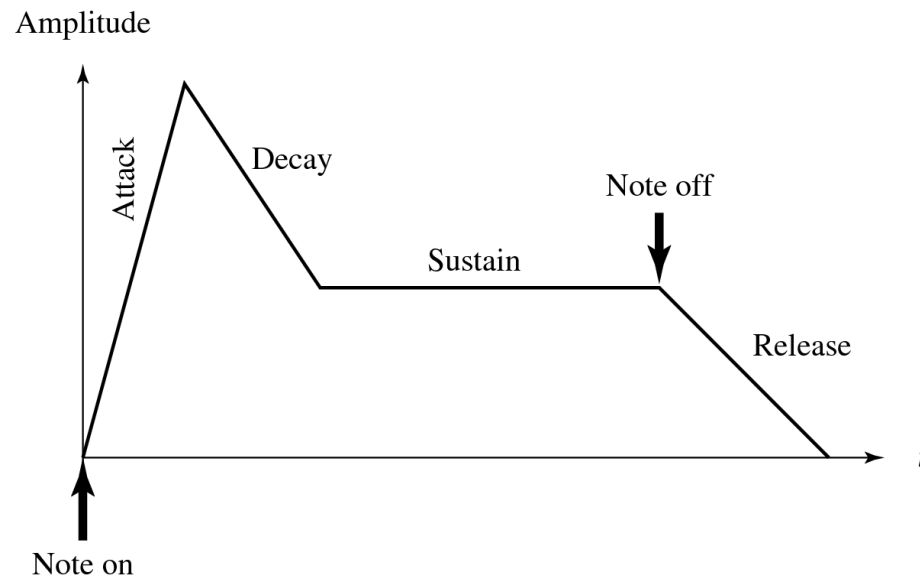


Fig. 6.9: Stages of amplitude versus time for a music note

## 6.3 Quantization and Transmission of Audio

- **Coding of Audio**

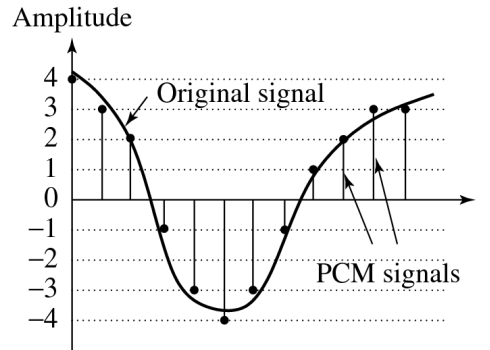
- a) Quantization – using a non-uniform quantization allows us to encode the signal better with fewer bits.
- b) Encoding Differences in signals between the present and a past time can reduce the size of signal values into a much smaller range.

- Every compression scheme has three stages:
  - A. The input data is **transformed** to a new representation that is easier or more efficient to compress.
  - B. We may introduce **loss** of information. ( e.g due to *Quantization*  $\Rightarrow$  we use a limited number of quantization levels, so noise is introduced).
  - C. **Coding**. Assign a codeword to each output level.

# Pulse Code Modulation (PCM)

- The basic techniques for creating digital signals from analog signals are **sampling** and **quantization**.
- Quantization consists of selecting breakpoints in magnitude, and then re-mapping any value within an interval to one of the representative output levels.

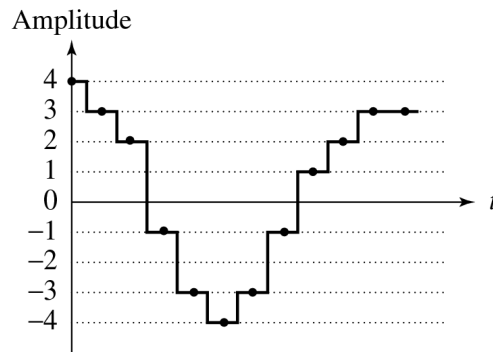
## 2. Decoded signal is discontinuous.



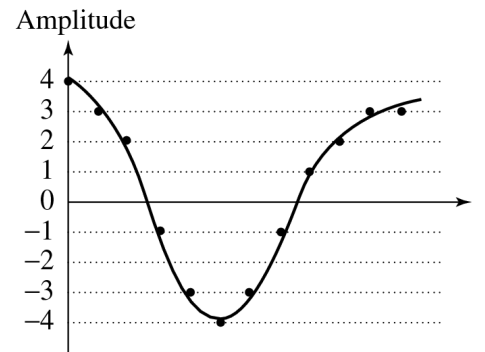
Sample points (PCM signals)

(a)

Signal decoded from sample points



(b)



Reconstructed signal after low-pass filtering

(c)

Fig. 6.13: Pulse Code Modulation (PCM). (a) Original analog signal and its corresponding PCM signals. (b) Decoded staircase signal. (c) Reconstructed signal after low-pass filtering.

# Differential Coding of Audio

- Audio is often stored not in simple PCM but instead in a form that exploits differences — which are generally smaller numbers, so offer the possibility of using fewer bits to store.
  - (a) One possibility for the assignment of codewords to differences is to assign short codes to frequently occurring values (small differences) and long codewords to rarely occurring ones (large differences).

# Lossless Predictive Coding

- **Predictive coding:** simply means transmitting differences — predict the next sample as being equal to the current sample; send not the sample itself but the difference between previous and next.
  - (a) Predictive coding consists of finding differences, and transmitting these using a PCM system.
  - (b) Note that differences of integers will be integers. Denote the integer input signal as the set of values  $f_n$ . Then we **predict** values  $\hat{f}_n$  as simply the previous value, and define the error  $e_n$  as the difference between the actual and the predicted signal:

$$\hat{f}_n = f_{n-1}$$

$$e_n = f_n - \hat{f}_n \tag{6.12}$$

(c) But it is often the case that some **function** of a few of the previous values,  $f_{n-1}$ ,  $f_{n-2}$ ,  $f_{n-3}$ , etc., provides a better prediction. Typically, a linear **predictor** function is used:

$$\hat{f}_n = \sum_{k=1}^{2 \text{ to } 4} a_{n-k} f_{n-k} \quad (6.13)$$

- *Lossless predictive coding* — As a simple example, suppose we devise a predictor for  $\hat{f}_n$  as follows:

$$\hat{f}_n = \left\lfloor \frac{1}{2} (f_{n-1} + f_{n-2}) \right\rfloor \quad (6.14)$$

$$e_n = f_n - \hat{f}_n$$

- Let's consider an explicit example. Suppose we wish to code the sequence  $f_1, f_2, f_3, f_4, f_5 = 21, 22, 27, 25, 22$ . For the purposes of the predictor, we'll invent an extra signal value equal to  $f_0 = 21$ , and first transmit this initial value, uncoded.

(6.15)

- Let's consider an explicit example. Suppose we wish to code the sequence  $f_1, f_2, f_3, f_4, f_5 = 21, 22, 27, 25, 22$ . For the purposes of the predictor, we'll invent an extra signal value equal to  $f_1 = 21$ , and first transmit this initial value, uncoded.

$$\hat{f}_2 = 21, e_2 = 22 - 21 = 1;$$

$$\hat{f}_3 = \left\lfloor \frac{1}{2}(f_2 + f_1) \right\rfloor = \left\lfloor \frac{1}{2}(22 + 21) \right\rfloor = 21,$$

$$e_3 = 27 - 21 = 6;$$

$$\hat{f}_4 = \left\lfloor \frac{1}{2}(f_3 + f_2) \right\rfloor = \left\lfloor \frac{1}{2}(27 + 22) \right\rfloor = 24,$$

$$e_4 = 25 - 24 = 1;$$

$$\hat{f}_5 = \left\lfloor \frac{1}{2}(f_4 + f_3) \right\rfloor = \left\lfloor \frac{1}{2}(25 + 27) \right\rfloor = 26,$$

(6.15)

$$e_5 = 22 - 26 = -4$$

- The error does center around zero, we see, and coding (assigning bit-string codewords) will be efficient.

How would you decode the transmitted signal?

- The error does center around zero, we see, and coding (assigning bit-string codewords) will be efficient.

How would you decode the transmitted signal?

You receive transmitted the errors  $e_n$

Then

$$\hat{f}_n = \left[ \frac{1}{2} (f_{n-1} + f_{n-2}) \right]$$

$$f_n = e_n + \hat{f}_n$$