Matlab 15: Audio Processing

Cheng-Hsin Hsu

National Tsing Hua University

Department of Computer Science

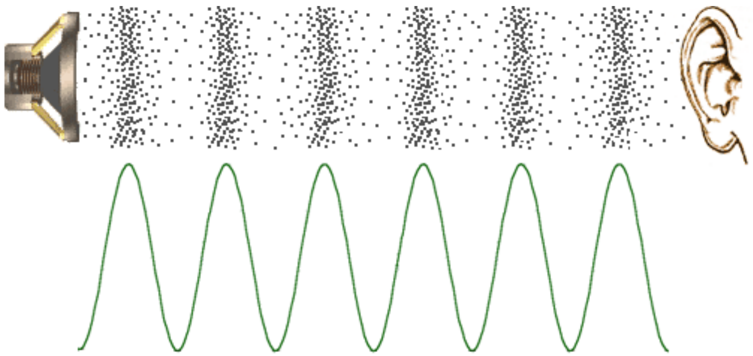
Slides and sample codes are based on the materials from
Prof. Roger Jang

What Are Audio Signals?

- Audio signals are…
  - Signals that are audible to human, such as speech and music
  - The range of fundamental frequencies of audible signals is about 20 ~ 20000 Hz.
    - The range is wider for the young people, narrower for the elderly.

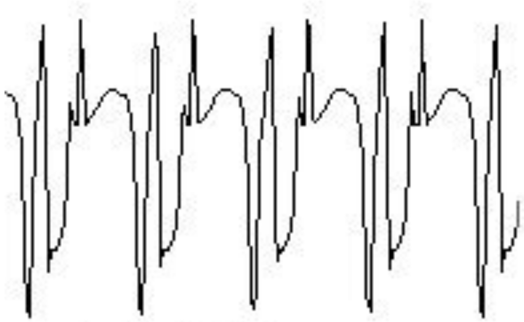
Voice Generation & Reception

- Steps in voice generation & reception
  - Vibration of voice source
  - Resonance by surrounding objects
  - Traveling through air (or other media)
  - Reception of membranes and neurons at inner ears
  - Recognition by brains
- Examples
  - Singing
  - Whistling
  - Guitar
  - Flute



Categorization of Audio Signals

- Number of sources
  - Monophonic
  - Polyphonic
- Waveform
  - Quasi-periodic sound
    - voiced sound of speech
  - Aperiodic sound
    - Unvoiced sound of speech
- Source types
  - Sounds from animals (bioacoustics)
    - Dog barking, cat meowing, frog croaking, duck quacking
  - Sounds from non-animals
    - Car engines, thunders, music instruments



periodic (pitch)

aperiodic (noise)

S/U/V in Speech

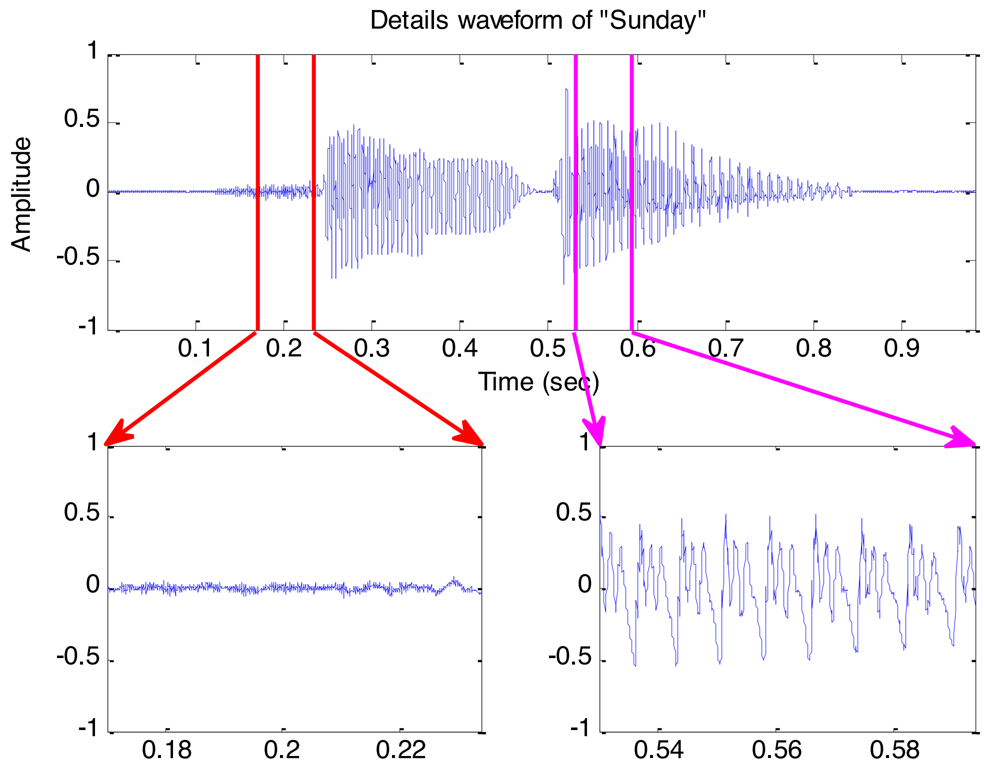
- Speech signals can be divided into S, U, V
  - S (silence): no speech activity
  - U (unvoiced): speech activity without vibration from vocal chords
  - V (voiced): speech activity with vibration
- How to detect S, U, V?
  - By putting your hand on your throat to feel the vibration
  - By waveform observation

Tools for General Audio Processing

- Tools for recording and waveform observation
  - Cool Edit
  - GoldWave
  - Audacity
  - MATLAB
- Quiz question!
  - What is the major difference between the waveforms of speech and whistle?

Speech Signal of “Sunday”

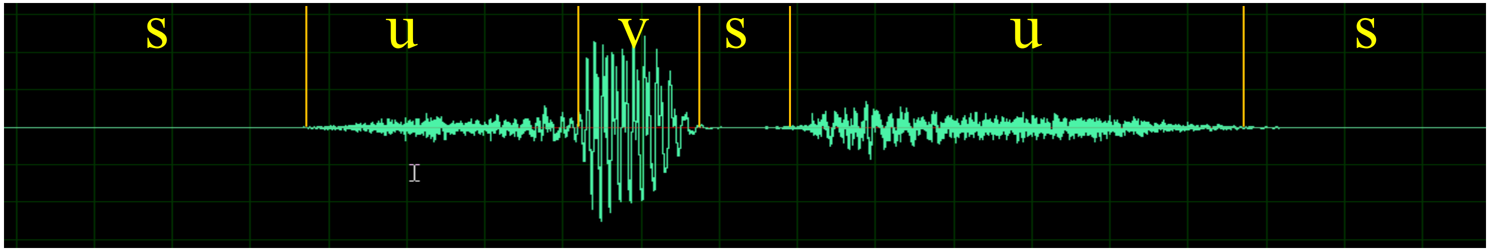
- Unvoiced vs. voiced frames



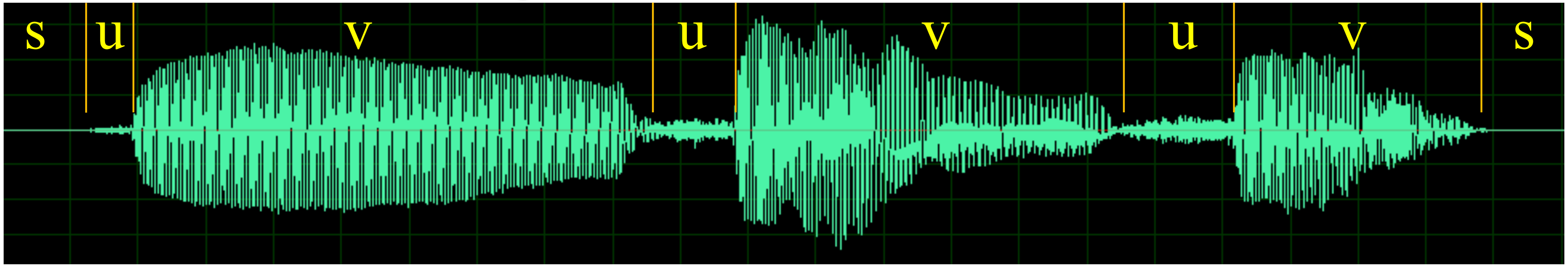
Silence, Unvoiced and Voiced Sounds

- Examples of S, U, V
  - "Six"

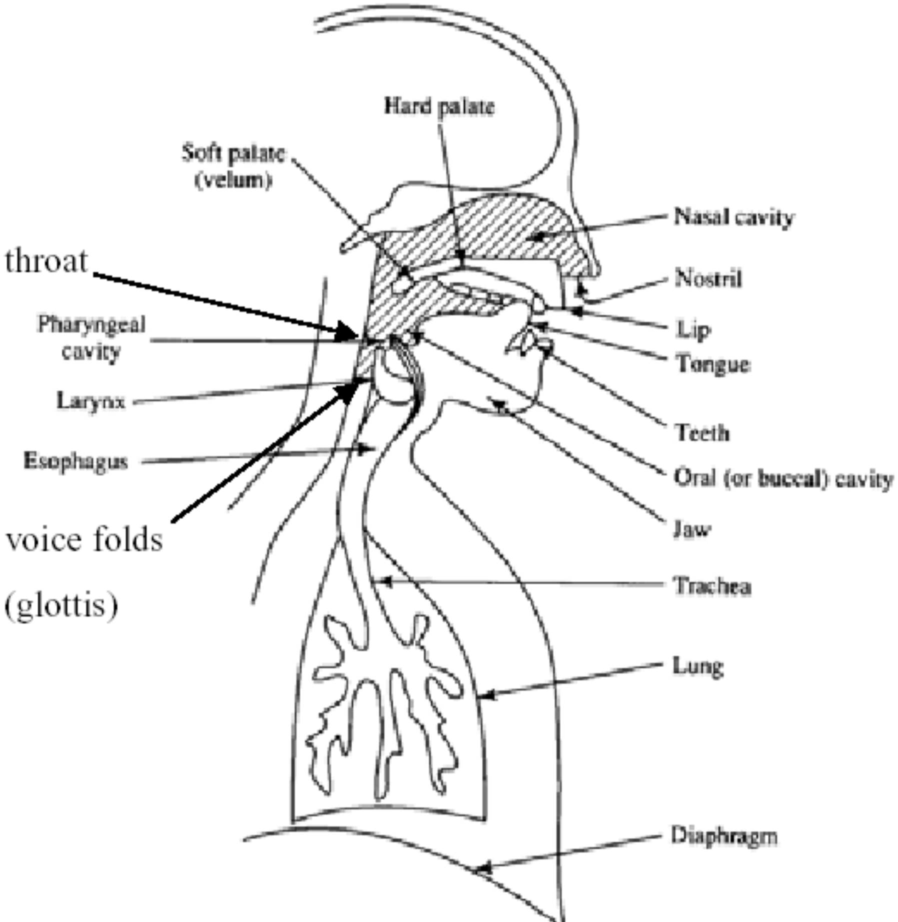
Quiz candidate!

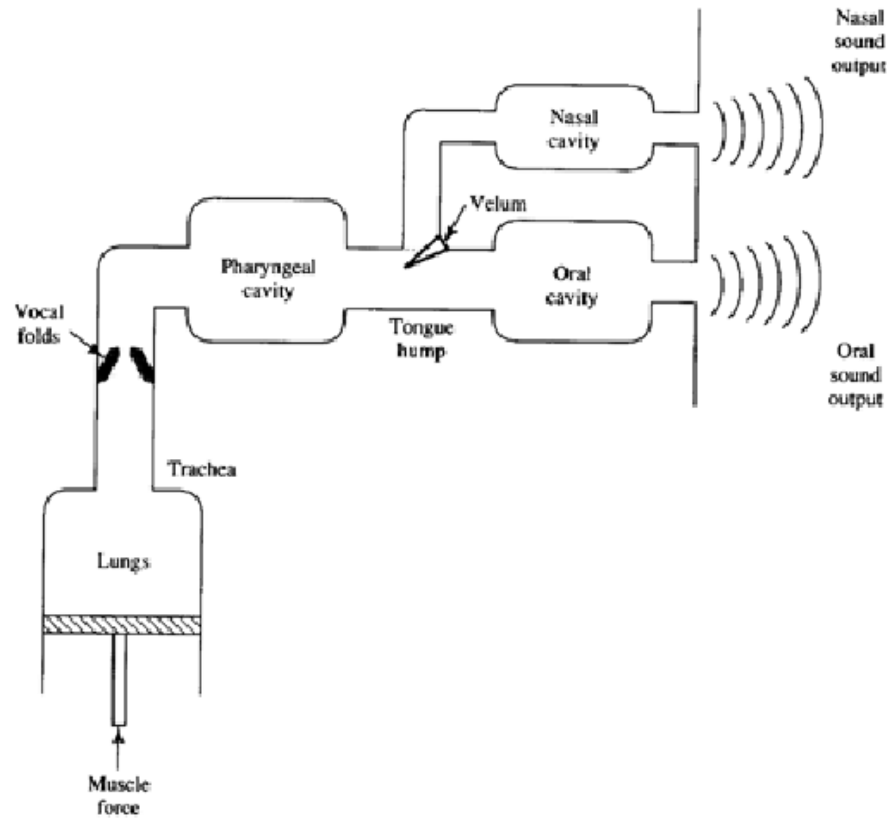


  - "資訊系"



Human Speech Production

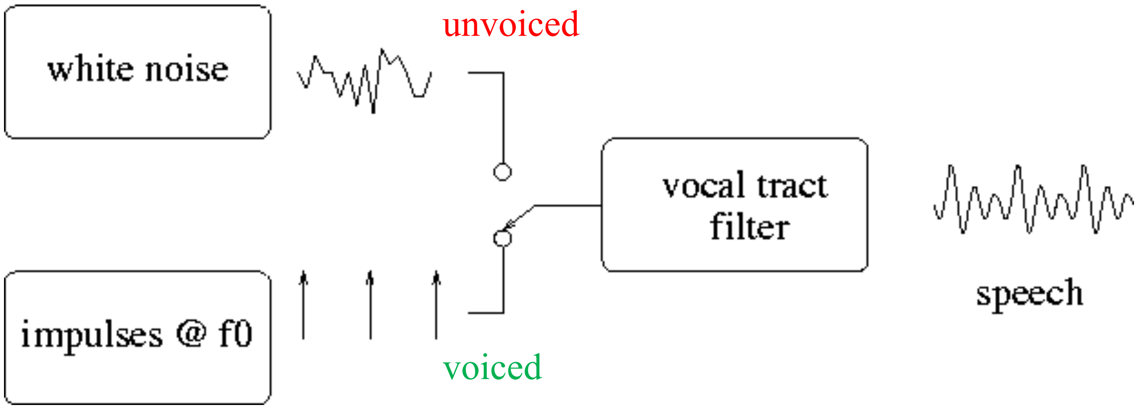




8 vocal + pharyngeal cavities = vocal tract MSP

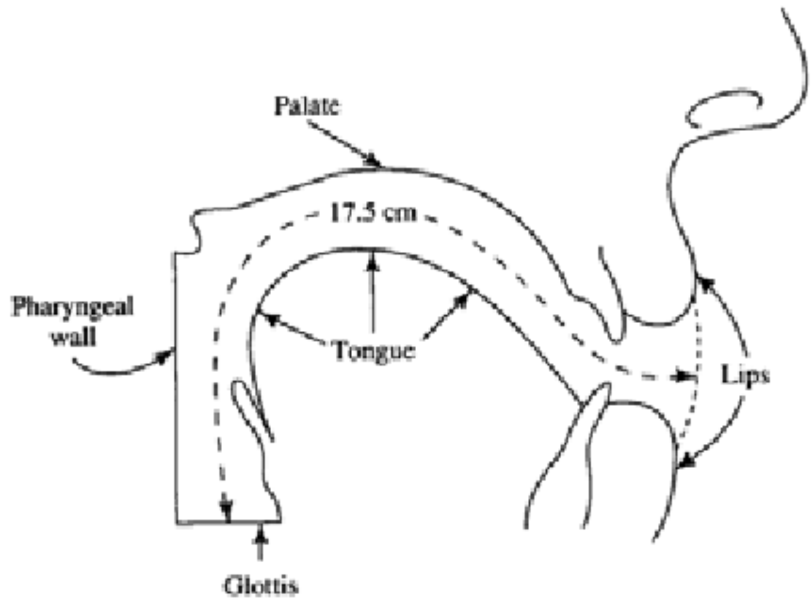
Source-filter Model for Human Speech Production

Speech is split into a rapidly varying excitation signal and a slowly varying filter. The envelope of the power spectra contains the vocal tract info.

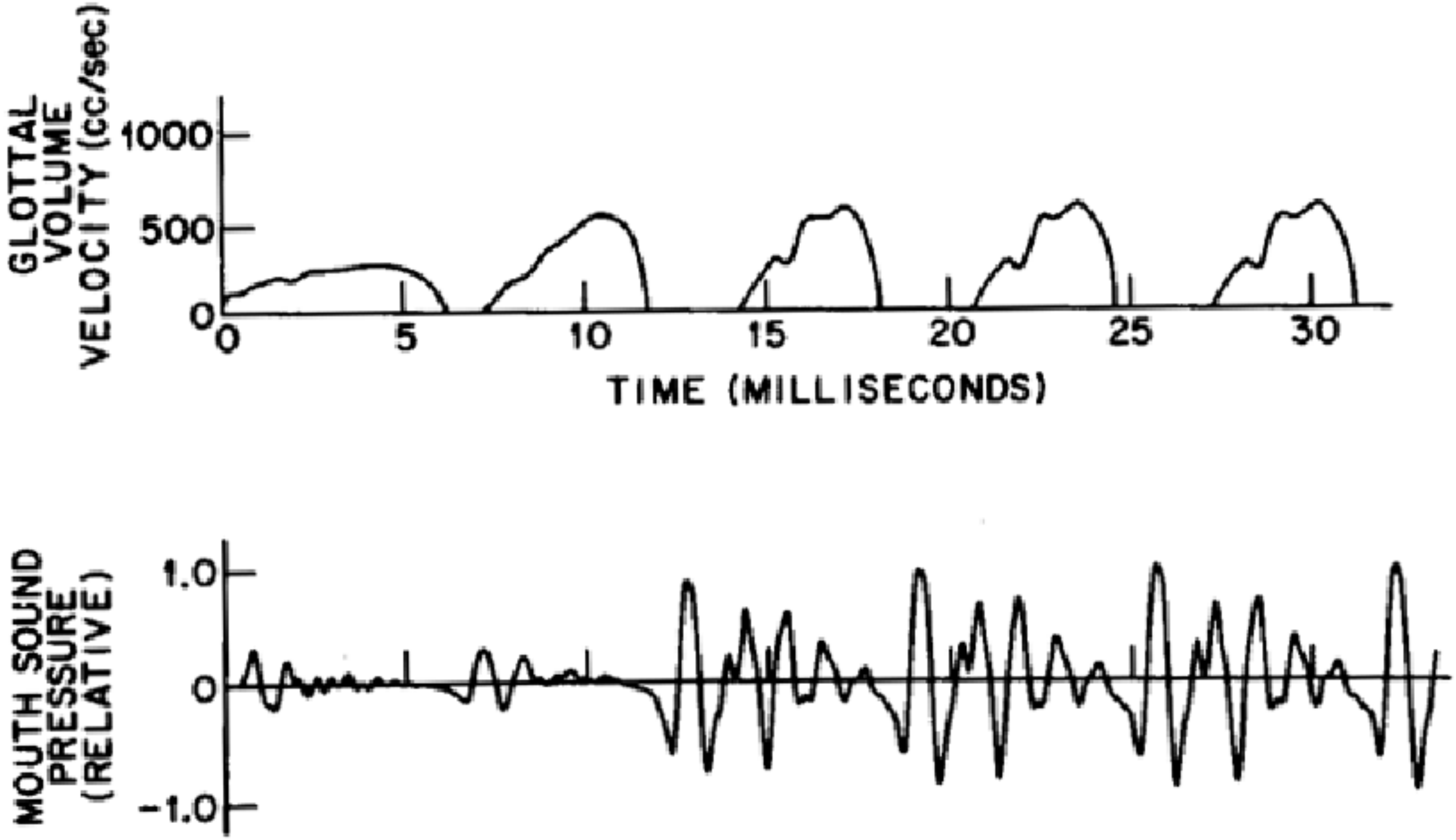


Two important characteristics of the model are fundamental frequency (f0) and formants (F1, F2, F3, …)

The Vocal Tract

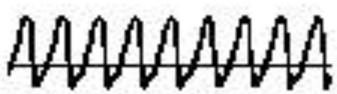


Glottal Volume Velocity & Resulting Sound Pressure (Voiced)



Speech Production

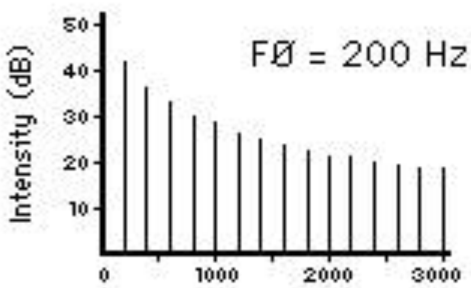
Glottal Pulses | **Vocal Tract** | **Speech Signal**

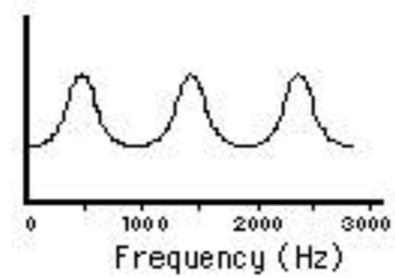


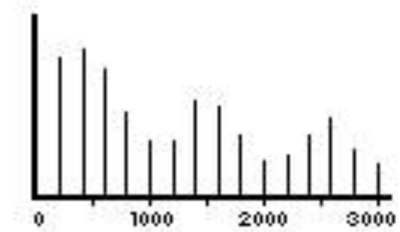
+

+

=







(a) Source Spectrum | **(b) Filter Function** | **(c) Output Energy Spectrum**

Videos for Vocal Cords Movement

- Movement of vocal cords
  - http://www.youtube.com/watch?v=mJedwz_r2Pc
  - http://www.youtube.com/watch?v=v9Wdf-RwLcs

Parameters for Recording

- Three major parameters for recording audio files
  - Sample rate: no. of samples per sec
    - 8 kHz (phone quality)
    - 16 KHz (for common speech recognition)
    - 44.1 KHz (CD quality)
  - Bit resolution: no. of bits for representing a sample
    - 8-bit (uint8 with range: 0~255)
    - 16-bit (int16 with range: -32768~32767)
  - No of channels
    - Mono
    - Stereo

Storage for Audio Files

- Examples of storage requirement
  - 1 min. of recording with fs=16000, nbits=16, #channel=1➔ 60 (sec)*16 (KHz)*2 (byetes)*1 (channel) = 1920 KB = 1.92 MB
  - 3-mins of CD music with fs=44.1KHz, nbits=16, #channel=2 ➔ 180 (sec)*44.1 (KHz)*2 (bytes)*2 (channels) = 31752 KB = 32 MB

How to Generate a Sine Wave Signal

- Math formula: $y = a * \sin(2\pi f t + \theta)$
- MATLAB code:

```
duration=3;
f=440;
fs=16000;
time=(0:duration*fs-1)/fs;
y=0.8*sin(2*pi*f*time);
plot(time, y);
sound(y, fs);
```

Analog & Discrete Phenomena

- Analog: continuous phenomenon, between any two points there exist infinite number of points
  - Most natural phenomena
- Discrete: points (either in time or space) are clearly separated
- Computers work with discrete values ➔ analog-to-digital conversion
- Digital media:
  - Better quality, less susceptible to noise
  - More compact to store and transmit (high compression ratios)

Analog-to-Digital Conversion: Two Steps

- Sampling:
  - choose discrete points at which we measure (sample) the continuous signal
  - Sampling rate is important to recreate the original signal
- Quantization:
  - Represent each sample using a fixed number of bits
  - Bit depth (or sample size) specifies the precision to represent a value

Sampling

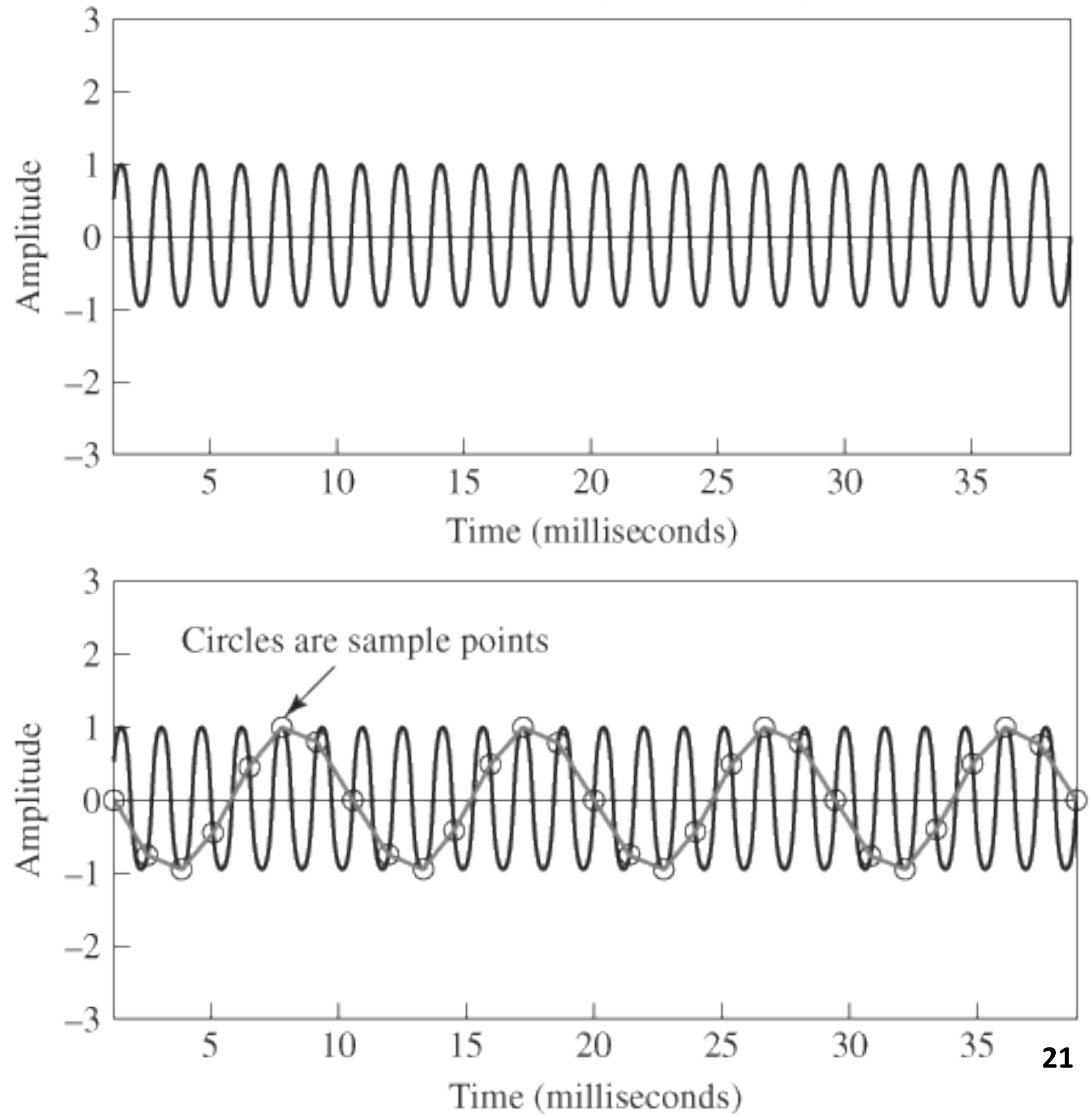
- Nyquist frequency
  - The minimum sampling rate to reconstruct the original signal: $r = 2f$
  - f is the frequency of the signal

- Under sampling can produce distorted/different signals (aliasing)

Under Sampling: Example

- *f = 637* Hz

- Sampling at *770 (< 2 f)* produces a different wave

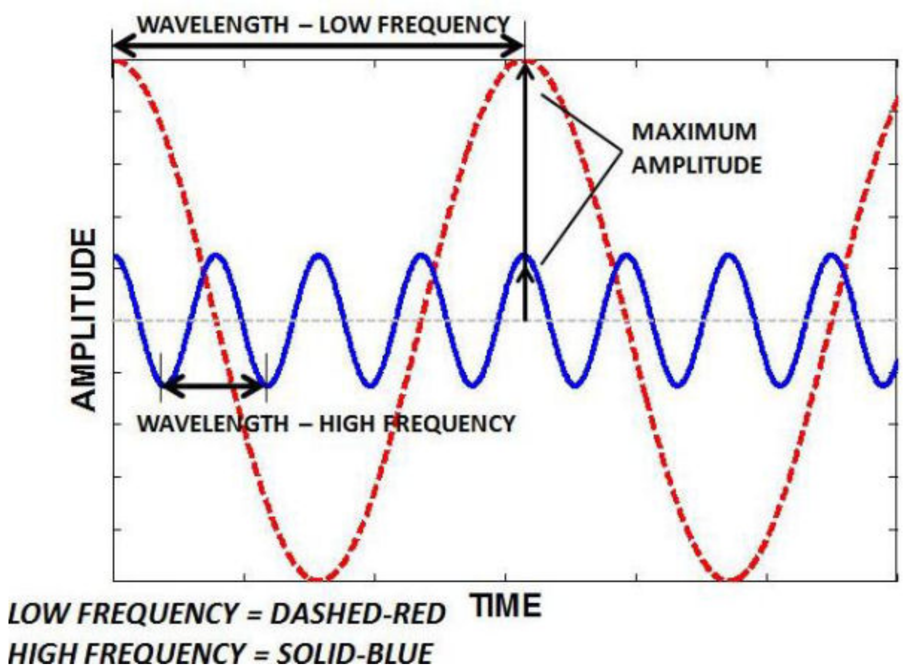


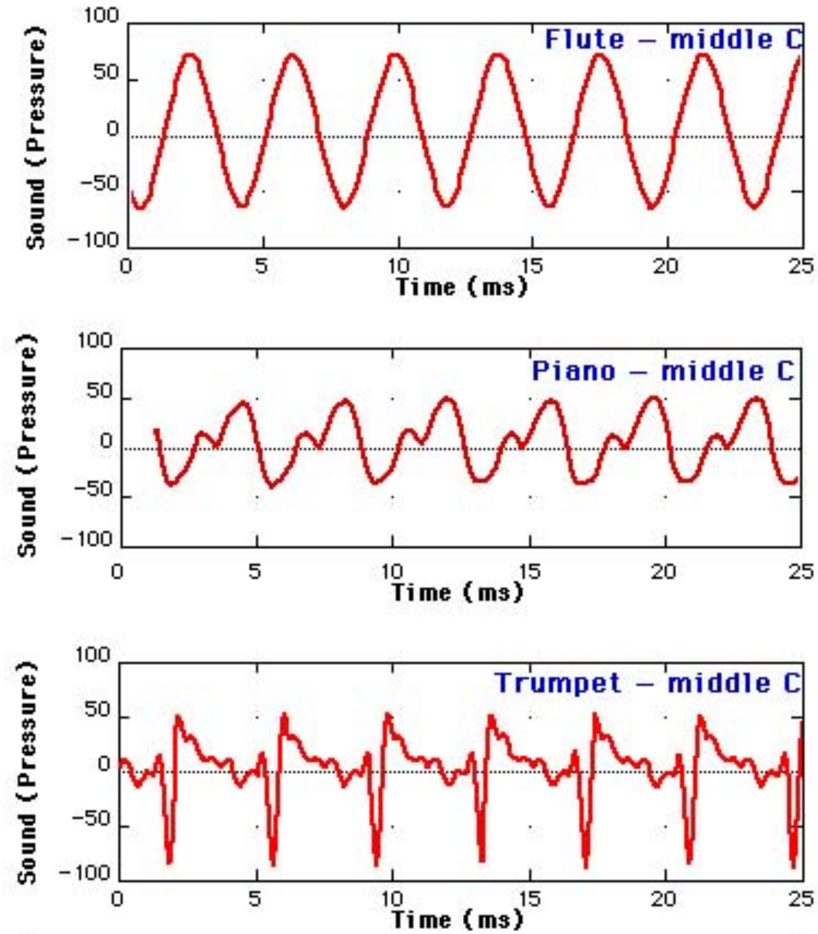
Quantization

- *n* bits to represent a digital sample ➔ max number of different levels is $m = 2^n$
- ➔ real, continuous, sample values are rounded (approximated) to the nearest levels
- ➔ Some information (precision) could be lost

Properties of Audio Signals

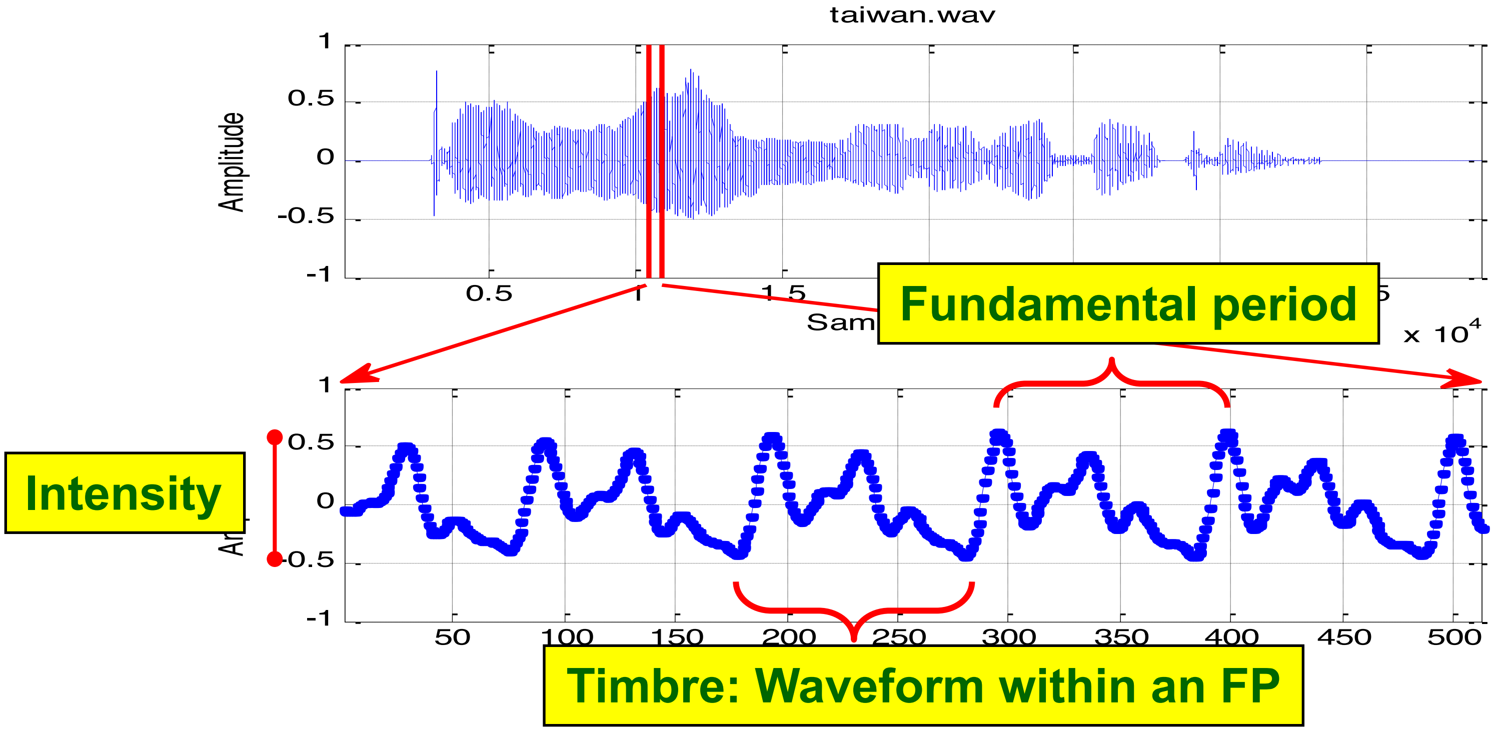
- Volume: amplitude, loudness, intensity, or energy
- Pitch: fundamental frequency
- Timbre: tone color or quality





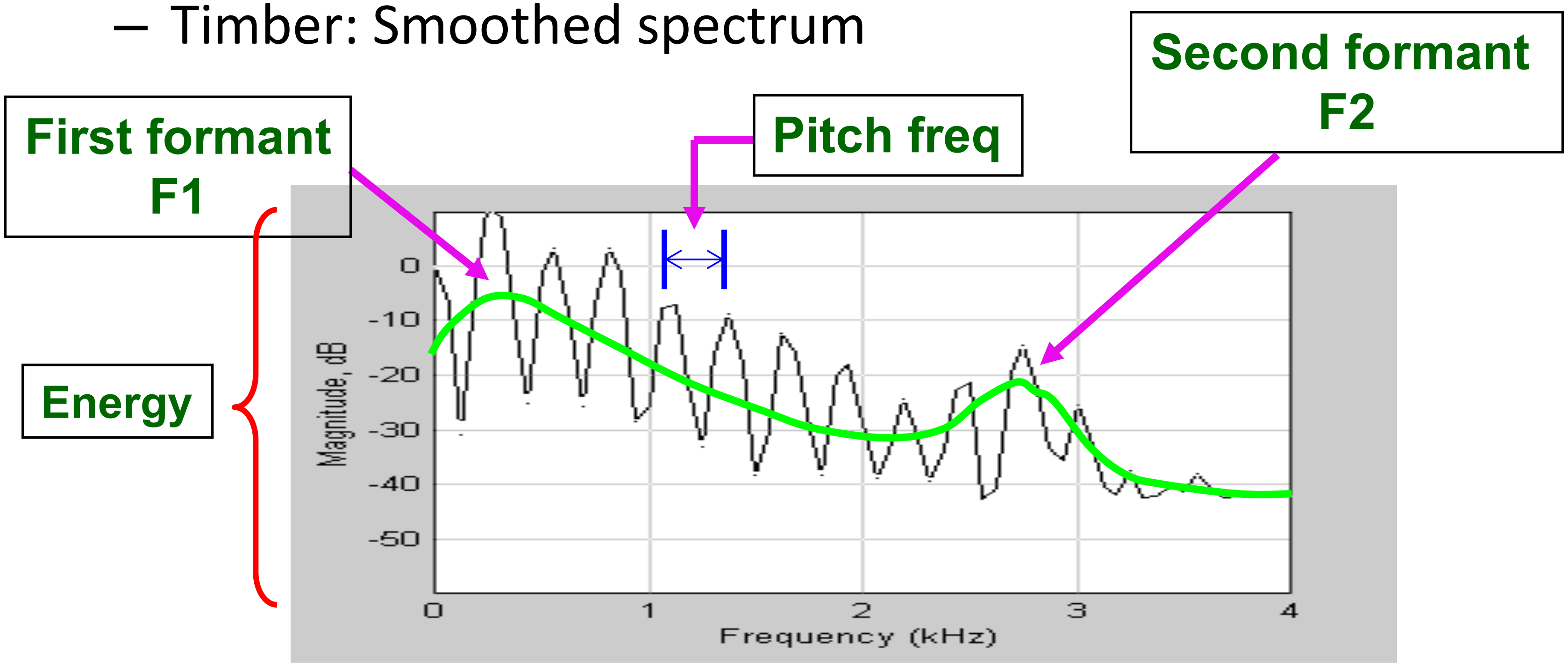
Time-domain Features

- Time-domain audio features presented in a frame (analysis window)



Frequency-domain Features

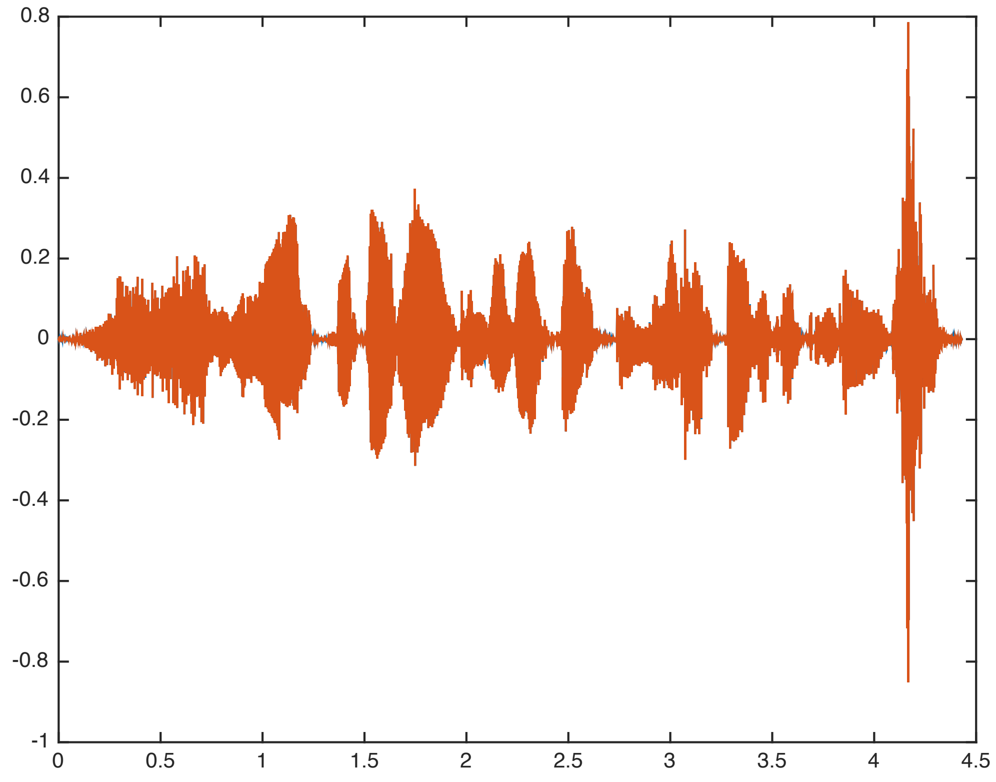
- Frequency-domain audio features in a frame
  - Energy: Sum of power spectrum
  - Pitch: Distance between harmonics
  - Timber: Smoothed spectrum



Read, Play, and Visualize Audio Files

- Use audioread to read a wav file
- Use sound to play the sound
- Plot the waveform

```
[y, fs]=audioread('bear.wav');
sound(y, fs);
time=(1:length(y))/fs;
plot(time, y);
```



Read the Metadata in Audio Files

- Reading metadata
  - info=audioInfo('file');
  - Different types of audio files may return different fields of info.
- Two types of reading data from audio files
  - For audio itself
    - y= audioread('file')
  - For metadata
    - info=audioinfo('file')

```
fileName='bear.wav';
info=audioinfo(fileName);
fprintf('Filename = %s\n', info.Filename);
fprintf('Compression = %s\n', info.CompressionMethod);
fprintf('No. Channels = %g\n', info.NumChannels);
fprintf('Smpling Rate = %g Hz\n', info.SampleRate);
fprintf('Samples = %g\n', info.TotalSamples);
fprintf('Duration = %g secs \n', info.Duration);
fprintf('Resolution = %g bits/sample\n', info.BitsPerSample);
```

Normalization on Audio Signals

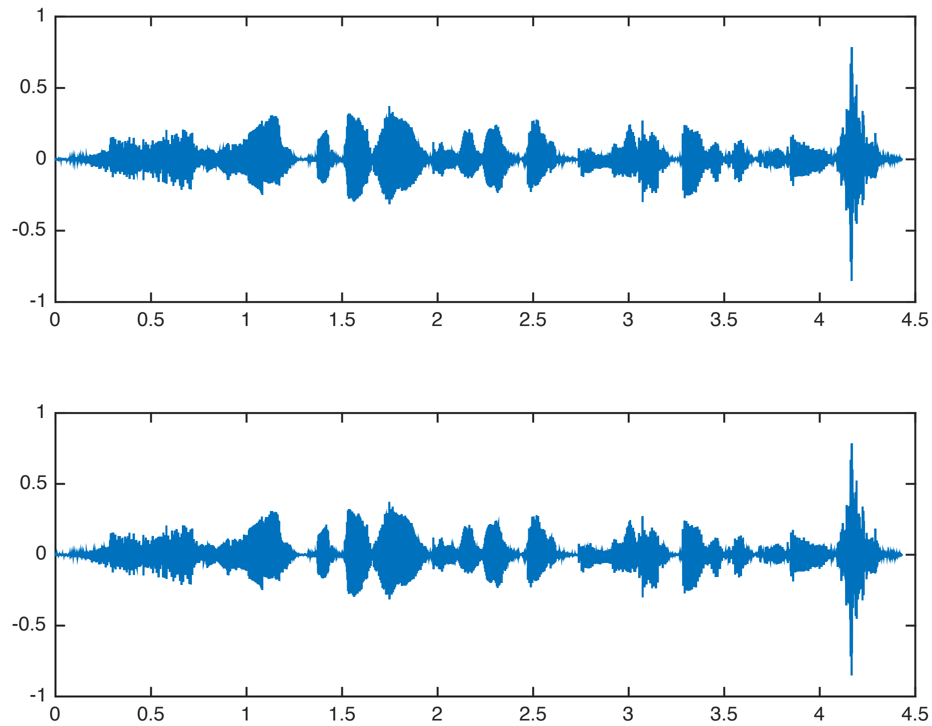
- Data is audio files
  - 8 bits ➔ uint8, $[0, 2^8-1]$
  - 16 bits ➔ int16, $[-2^{15}, 2^{15}-1]$
- MATLAB's method to scale to range [-1, 1]
  - 8 bits ➔ (y-128)/128
  - 16 bits ➔ y/32768
- Check MATLABs' scaling

```
fileName='bear.wav';
[y, fs]=audioread(fileName);
info=audioinfo(fileName);
nbits=info.BitsPerSample;
y0=round(y*32768)
```

Stereo Audio Files

- audioread can also read stereo wav files
- Each column represents a L-R channel

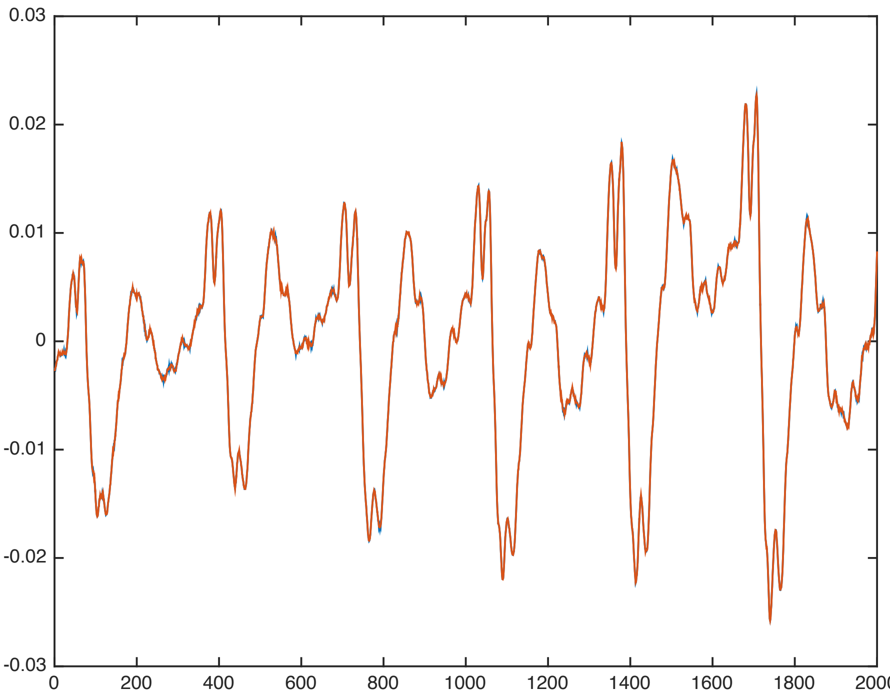
```
fileName= 'bear.wav';
[y, fs]=audioread(fileName);
sound(y, fs);
left=y(:,1);
right=y(:,2);
subplot(2,1,1), plot((1:length(left))/fs, left);
subplot(2,1,2), plot((1:length(right))/fs, right);
```



Read Part of an Audio File

– If we only want to read parts of an audio file
– audioRead05.m

```
[y,fs]=audioread('bear.wav', [5001 7000]);     figure; plot(y)
```



Play Audio in Matlab

- After reading audio files into Matlab
- We can also process the audio data
  - Increase/decrease volume
  - increase/reduce pitches
  - De-noise
- Ant then play out the result audio signals

Play Audio (1/2)

- Play a single sound

```
load bear.mat
sound(y, fs);
```

- Play multiple sounds

```
[y, fs]=audioread('bear.wav');
sound(y, fs);
pause(1);
sound(3*y, fs);
pause(1);
sound(3*y, fs*0.8);
```

Play Audio (2/2)

- Create a audio object
  - audioplayer
  - play
  - playblocking

- Play a single sound

```
load bear.mat
p=audioplayer(y, fs);
play(p);
```

- Sequentially Play multiple sounds

```
[y, fs]=audioread('bear.wav');
p=audioplayer(y, fs);
playblocking(p);
load bear.mat
p=audioplayer(y, fs * 1.2);
playblocking(p);
```

Changing the Amplitudes

- Adjust volumes
- Question: do you think the volume goes up for 3, 5, and 7 times in the following example?

```
[y, fs]=audioread('bear.wav');
p=audioplayer(1*y, fs); playblocking(p);
p=audioplayer(3*y, fs); playblocking(p);
p=audioplayer(5*y, fs); playblocking(p);
p=audioplayer(7*y, fs); playblocking(p);
```

Changing the Sampling Rates (1/2)

- New sampling rates → new play duration *and* new pitches
- In the following example, the play durations get shorter and the pitches go higher → sounds like Donald Duck

```
[y, fs]=audioread('bear.wav');
p=audioplayer(y, fs);
p.SampleRate=1.0*fs; playblocking(p);
p.SampleRate=1.2*fs; playblocking(p);
p.SampleRate=1.5*fs; playblocking(p);
p.SampleRate=2.0*fs; playblocking(p);
```

Changing the Sampling Rates (2/2)

– In the following example, the play durations get longer and the pitches go lower→ sounds like cows

```
[y, fs]=audioread('bear.wav');
p=audioplayer(y, fs);
p.SampleRate=1.0*fs; playblocking(p);
p.SampleRate=0.8*fs; playblocking(p);
p.SampleRate=0.6*fs; playblocking(p);
p.SampleRate=0.4*fs; playblocking(p);
```

Observations

- Observations
  - Higher sample rate for playback leads to…
    - Shorter duration and higher pitch
  - Lower sample rate for playback leads to...
    - Longer duration and lower pitch
- How to…
  - Generate higher pitch without duration change?
    - ➔ Pitch modification
  - Create longer duration without pitch change?
    - ➔ Duration modification

Change the Audio Signals

- 0) Play the wav as-is
- 1) Change the sign of audio signals
- 2) Reverse the signals (along the time domain)
- What will happen?

```
[y, fs]=audioread('bear.wav');
p=audioplayer(y, fs); playblocking(p);
p=audioplayer(-y, fs); playblocking(p);
p=audioplayer(flipud(y), fs); playblocking(p);
```

Volume Adjustment

– Soundsc scales the data so that the sound is played as loud as possible without clipping

```
[y, fs]=audioread('bear.wav');
sound(y, fs);
fprintf('Press any key to continue...\n'); pause
soundsc(y, fs);
```

Record Audio Files

- We have seen how to read audio files
- We have learned how to play audio files
- Let’s create new audio files
  - audiorecorder
  - recordblocking

Audio Recording Example (1/2)

- Record 3 seconds using default settings

```
duration=3;
recObj=audiorecorder;
recordblocking(recObj, duration);
fprintf('Press any key to play out : '); pause
play(recObj);
```

  - Default settings
    - Sampling rate: 8000 Hz
    - Per sample resolution: 8 bits
    - Mono

Audio Recording Example (2/2)

- Use non-default settings

```
fs=16000;       % sampling rate
nBits=16;       % bit resolution
nChannel=1;     % no. channel
duration=3;     % duration in seconds
recObj=audiorecorder(fs, nBits, nChannel);
fprintf('Press any key to start recording for %g seconds：', duration);
pause
fprintf('recording...');
recordblocking(recObj, duration);
fprintf('Press any key to playout...'); pause
play(recObj);
y = getaudiodata(recObj, 'double'); % get data as a double array
plot((1:length(y))/fs, y);
xlabel('Time (sec)'); ylabel('Amplitude');
```

Write Audio Records as Files (1/2)

- Matlab also allows us to save recordings as files
  - audiowrite(audioFile, y, fs)
    - audioFile is the filename，y is the audio sample，fs is the sampling rate

```
load bear.mat
audioFile='bear2.wav';
fprintf('Saving to %s...\n', audioFile);
audiowrite(audioFile, y, round(1.5*fs));
fprintf('Press any key to play %s...\n', audioFile);
dos(['open ', audioFile]);
```

Write Audio Records as Files (2/2)

- Combine recording, playing, and saving into the following code

```
fs=16000;
nBits=16;
nChannel=1;
duration=3;
recObj=audiorecorder(fs, nBits, nChannel);
fprintf('Press any key to record for %g seconds : ', duration); pause
recordblocking(recObj, duration);
y = getaudiodata(recObj, 'double');
plot((1:length(y))/fs, y); xlabel('Time (sec)'); ylabel('Amplitude');
sound(y, fs);
audioFile='myRecording.wav';
fprintf('Saving to %s...\n', audioFile);
audiowrite(audioFile, y, fs);
system('open myRecording.wav');
```

Matlab #13 Homework (M13)

1. Wave recording (2%): Write a MATLAB script recordMyVoice01.m to record 10 seconds of your speech, say introduce yourself. Save your recording as myVoice.wav. Other recording parameters are: sample rate = 16 KHz, bit resolution = 16 bits. Please use the script to print out answers to the following questions within the MATLAB window.
   - How much space is taken by the audio data in the MATLAB workspace?
   - What the data type of the audio data?
   - How do you compute the amount of the required memory from the recording parameters?
   - What is the size of myVoice.wav?
   - How many bytes is used in myVoice.wav to record overheads other than the audio data itself?