



COGS 119/219

MATLAB for Experimental Research

Fall 2014 – Week 5

Sounds in Matlab

Sounds

- Sounds are also vectors like images.
- Pure tones are sine waves.
- The volume of the sound depends on the **amplitude**, and the pitch of the tone depends on the **frequency**.
- Pure tones are thus represented by functions of the form:

$$f(t) = A \sin(2 \pi \text{ freq } t)$$

where t is time.

- This family of functions has two parameters that we can hear **freq**, the frequency and **A**, the amplitude.

Sound basics

```
>> freq=500;      % frequency of the tone (Hz)
>> dur=1.5;       % duration of the tone (seconds)
>> sampRate=44000; % sampling rate
>> nTimeSamples = dur*sampRate; % number of time
                                % samples

>> t = linspace(0,dur,nTimeSamples);
>> y = sin(2*pi*freq*t);
>> sound(y,sampRate); % this plays the sound
```

```
>> help linspace
```

```
>> help sound
```

Sound basics

```
>> amp = 2;  
>> y2 = amp * y;  
>> sound(y2, sampRate); % this plays the  
                          % sound
```

Sound basics

```
>> noise = 0.2 * randn(1,sampRate);  
>> sound (noise, sampRate);
```

Plot

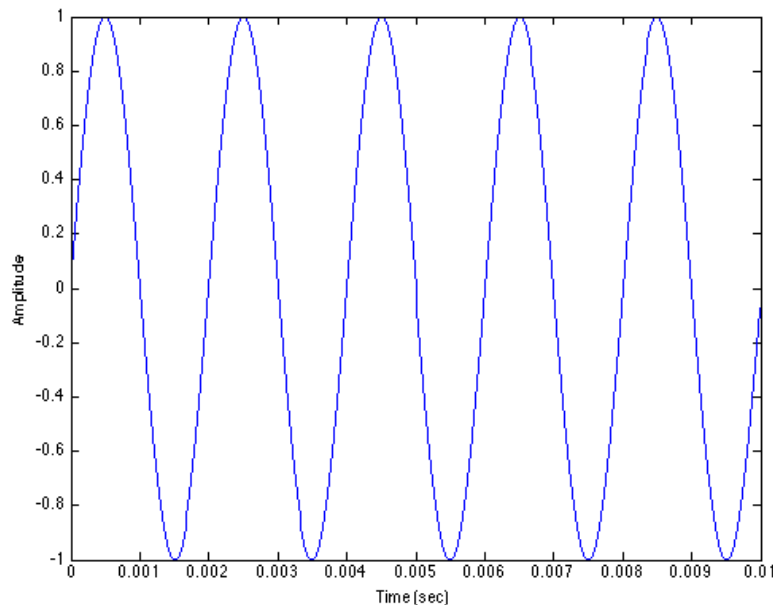
% lets look at the wave we created

```
>> indexToPlot = t<10/1000; % select only the  
                             % first 10 msec
```

```
>> plot(t(indexToPlot), y(indexToPlot));
```

```
>> xlabel('Time (sec)')
```

```
>> ylabel('Amplitude');
```



Make sound louder over time

`% Now lets make it louder over time`

```
>> ramp = linspace(0, 1, nTimeSamples); % create  
                                         % ramp
```

```
>> figure(2);
```

```
>> plot(t, ramp, 'r');
```

```
>> xlabel('Time (sec)');
```

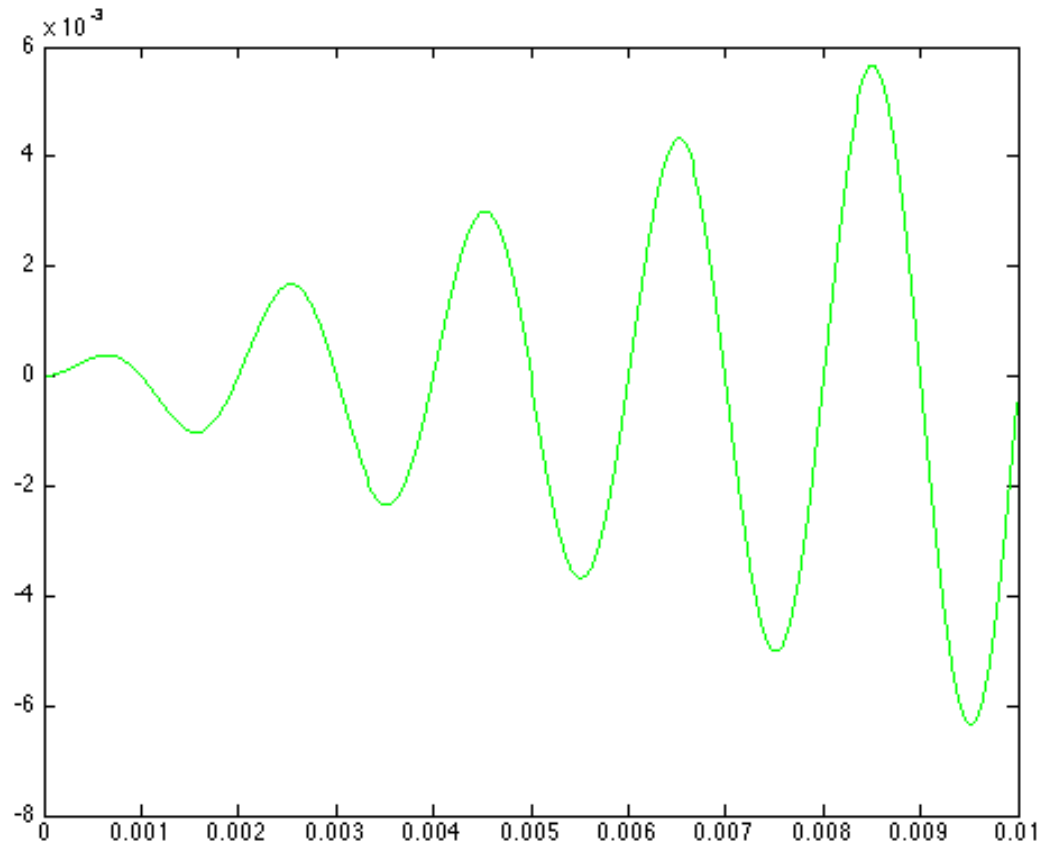
```
>> ylabel('amplitude envelope');
```

```
>> y3=y.*ramp; % create new sound with pure tone  
               % and amplitude ramp
```

```
>> sound(y3, sampRate);
```

Make sound louder over time

```
>> plot(t(indexToPlot), y3(indexToPlot), 'g');
```



What happens?

```
>> y4 = fliplr(y3);  
>> sound(y4, sampRate);
```

Exercise: Go over sounds_basic.m on the website.

Let's play a scale

- Frequencies of subsequent notes on the 12-tone scale are obtained by multiplying the previous frequency by $2^{(1/12)}$.
- Example:
$$\text{freqD} = \text{freqC} * (2^{(1/12)})$$
$$\text{freqE} = \text{freqD} * (2^{(1/12)}) = \text{freqC} * (2^{(2/12)})...$$
- How can we play a scale using `sound()`?

Let's play a scale: PlayNotes.m

```
freqC = 278.4375; % Frequency of middle C (Hz) (called 'C4')
numNotes=13;      % number of notes (13 notes = one octave (C to C))
noteNumbers = [0:(numNotes-1)];
whiteKeys = [1,3,5,6,8,10,12,13]; %white keys on piano (key of C)

% Frequencies of subsequent notes on the 12-tone scale
% are obtained by multiplying the previous frequency by
% 2^(1/12).
% Example: freqD = freqC*(2^(1/12))
% freqE = freqD*(2^(1/12)) = freqC*(2^(2/12))...

multFac = 2.^(noteNumbers/12);
allFreqs = freqC*multFac;

% Now, let's make sound waves containing tones for each note
dur = .8; %duration of the tone (Seconds)
ISI = 1; %time between the start of each note (Seconds)
sampRate = 8192; % Sampling rate for sound (Hz)
nTimeSamples = dur*sampRate; % number of time samples
t = linspace(0,dur,nTimeSamples);

% plays the scale
tic
for i=1:length(whiteKeys)
    freq = allFreqs(whiteKeys(i));
    y= sin(2*pi*freq*t);

    sound(y,sampRate);
    while toc<i*ISI
        % just keeping time
    end % end of the while loop
end

toc
```

Two speakers separately

```
% now lets do stereo
% you will need headphones to hear this
% we control the left and right speakers separately
% using a n x 2 (two column) matrix instead of a simple 1d vector
% In sndmat below, column 1 are for the left speaker
% column 2 for the right speaker.

] for i=1:length(whiteKeys)
    freq = allFreqs(whiteKeys(i));
    y= sin(2*pi*freq*t)';
    sndmat=repmat(y, 1, 2); % we simply replicated y
    if mod(i, 2)==0 % see help mod, basically odd or even
        sndmat(:, 1)=0; % zero one channel
    else
        sndmat(:, 2)=0; % zero the other channel
    end
    sound(sndmat, sampRate);
] while toc<i*ISI
-   end % end of the while loop
- end
```

Reading sounds

- Like images, we can read in sounds.

```
>> help wavread
```

`[Y,FS,NBITS] = WAVREAD(FILE)` returns the sample rate (FS) in Hertz and the number of bits per sample (NBITS) used to encode the data in the file.

Wavread

```
>> [cow,fs,nbits] = wavread('cowmoo.wav');
```

Reads in the sound. Check variables with whos

```
>> tic;
```

```
>> sound (cow,fs,nbits);
```

```
>> toc;
```

Plays the sound

```
>> soundsc (cow,fs,nbits);
```

Instead of sounds, you can also use soundsc, similar to imagesc

What does this do?

```
i=1;  
while cow(i) == 0  
    i=i+1;  
end;
```

```
j=size(cow,1);  
while cow(j) ==0  
    j=j-1;  
end;
```



```
>> [cow,fs,nbits] = wavread('cowmoo.wav');
```

Let's clip off silence from the beginning and end.

```
% find where sound starts
```

```
i=1;
```

```
while cow(i) == 0
```

```
    i=i+1;
```

```
end;
```




```
% find where sound ends
```

```
j=size(cow,1);
```

```
while cow(j) ==0
```

```
    j=j-1;
```

```
end;
```

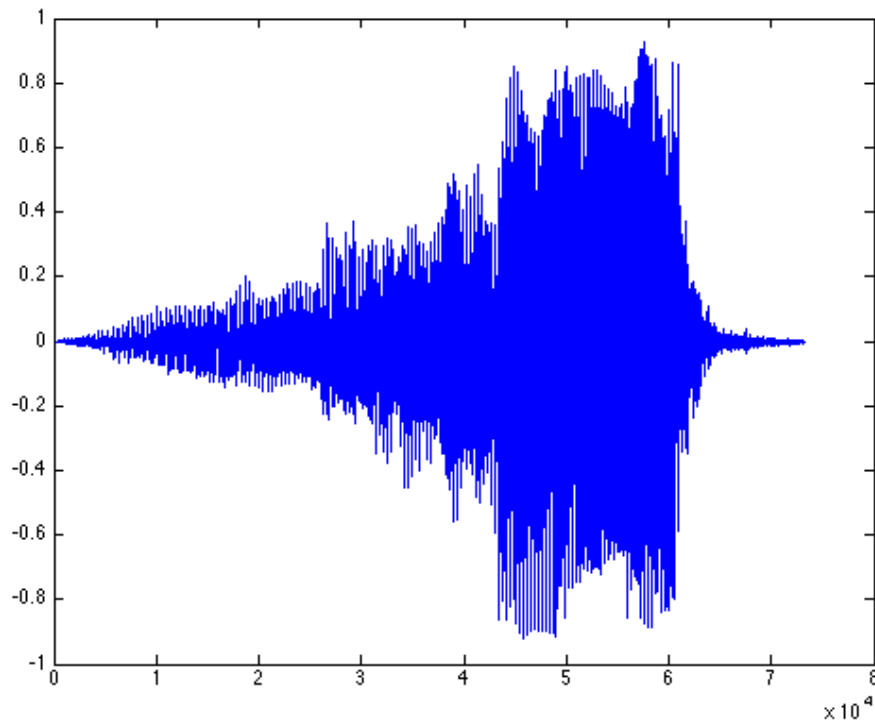
```
% select the middle where cow has nonzero values
```

```
cow2 = cow(i:j);
```

% look at the sound wave

```
x=1:size(cow2,1);
```

```
plot(x,cow2);
```





Let's play the sound backwards

Let's play the sound backwards

```
>> cwr = flipud (c);  
>> sound(cwr,fs,nbits);
```


Let's play the sound backwards

```
>> cwr = flipud (c2);
```

```
>> sound(cwr,fs,nbits);
```

% write the reversed sound into a new file


```
>> wavwrite (cwr,fs,'cowmoorev.wav');
```

- 
- Let's play cow sound and its reversed version together (e.g. as one file). How can we do that?

- Let's play cow sound and its reversed version together (e.g. as one file). How can we do that?

```
>> twocows = [cow2 + cowr];
```

```
>> sound(twocows, fs, nbits);
```



Let's play two cow sounds with 0.2 second delay in between the two speakers. How can we do that?

Let's play two cow sounds with 0.2 second delay in between the two speakers. How can we do that?

```
% add a delay to one sound by padding with zeros from start  
% we are calling them left and right because we will combine them  
% to create a stereo sound  
% if you use a 2 column matrix as input this will use stereo  
% see also PlayTones.m at the bottom where we use this approach
```

```
>> delay = 0.2;
```

```
>> left = cow2;
```

```
>> delaystart = round (delay * fs);
```

```
>> right = zeros (size(left));
```

```
>> right(delaystart:end) = left(1: (end-delaystart+1));
```

```
>> stereocow = [left right];
```

```
>> soundsc(stereocow,fs,nbits);
```

Alternative way to play sounds

```
>> tic;  
>> snd('Play','stereocow',fs);  
>> toc;
```

% alternative way to play sound using Psychtoolbox