AEGIS: Applications of Images and Signals in High Schools

Speech Processing - Unit Plan

J. Rebecca Dowell
RET Teacher
Titusville High School, Brevard Public Schools
dowell.jeanie@brevardschools.org

Dr. VetonKëpuska
Faculty Mentor
Florida Institute of Technology
vkepuska@fit.edu

Jacob Zurasky
Graduate Student Mentor
Florida Institute of Technology
jzurasky@my.fit.edu

July 20, 2012

Revised April 19, 2013

Executive Summary

Unit Introduction. As a homework assignment prior to beginning the unit, have each student do an Internet search on applications of speech processing. Introduce the unit by having each student share and discuss three examples of speech processing applications with the class. Demonstrate speech recognition software on a smart phone, and ask students how they think such software works. Next, demonstrate the SASE lab software.

Lesson 1: The Sound of a Sine Wave. Introduce modeling sound as a sinusoidal function. Students write MATLAB program to graph and play the sound of sine waves and the composite of sine waves with given frequencies. Concepts covered include: frequency, continuous vs. discrete functions, sampling rate, composite signals and function addition.

Lesson 2: Frequency Analysis. Introduce the Fourier Transform to transform a function from time domain to frequency domain. Students write MATLAB program to model a harmonic signal with a sum of sinusoidal functions, compute the FFT of the composite signal, and plot and analyze the frequency spectrum. Concepts covered include: harmonic signals modeled as a series of sinusoids, sine wave decomposition, Fourier Transform, and Euler's Formula.

Lesson 3: Digital Sound Effects. Introduce time-delay based sound effects. Students write MATLAB program to read a wave file and modify the signal with an echo sound effect. Concepts covered include: discrete functions, time-delay functions, and function operations.

Lesson 4: SASE Lab.Studentsparticipate in guided inquiry of SASE Lab software.

Unit Conclusion. Students conclude the unit by summarizing and reflecting on the unit in a presentation and report or poster.

Table of Contents

Lesson Plan 1: The Sound of a Sine Wave

NAMES: J. Rebecca Dowell and Jacob Zurasky

CONTENT AREA: Pre-Calculus

GRADE LEVEL: 9-12

TOPIC: Modeling Sound with Trigonometric Functions

NEXT GENERATION SUNSHINE STATE STANDARDS:

MA.912.A.2.2 Interpret a graph representing a real-world situation.

MA.912.A.2.7 Perform operations (addition, subtraction, division and multiplication) of functions algebraically, numerically, and graphically.

MA.912.A.2.13 Solve real-world problems involving relations and functions.

MA.912.D.11.3 Find specified terms of arithmetic and geometric sequences.

MA.912.T.1.6 Define and graph trigonometric functions using domain, range, intercepts, period, amplitude, phase shift, vertical shift, and asymptotes with and without the use of graphing technology.

MA.912.T.1.8 Solve real-world problems involving applications of trigonometric functions using graphing technology when appropriate.

COMMON CORE STATE STANDARD MATHEMATICS:

MACC.912.F-BF.1 Write a function that describes a relationship between two quantities.

MACC.912.F-BF.3 Identify the effect on the graph by replacing f(x) by f(x)+k, kf(x), f(kx), and f(x+k).

MACC.912.F-IF.3 Recognize that sequences are functions, sometimes defined recursively, whose domain is a subset of the integers.

MACC.912.F-IF.4 For a function that models a relationship between two quantities, interpret key features of graphs and tables in terms of the quantities.

MACC.912.F-TF.5 Choose trigonometric functions to model periodic phenomena with specified amplitude, frequency, and midline.

COMMON CORE MATHEMATICAL PRACTICE:
Model with mathematics.
Use appropriate tools strategically.

UNIT: Speech Processing

GOAL: Student will understand sound modeled as a sinusoidal function. Student will understand how changing parameters of a sine function affect the graph and the sound of the function. Student will understand the difference between a discrete and continuous function.

OBJECTIVE: Student will write continuous and discrete representations of a sine function with given parameters. Using technology, student will graph and play the sound of a sine function with a given frequency. Student will determine how changes in frequency affect the graphs and the sounds of sine functions. Student will add together sine functions and compare the results of the addition numerically and graphically. Students will evaluate a geometric sequence to determine the frequencies of musical notes.

MATERIALS: Computer with sound system. MATLAB software.

PRIOR KNOWLEDGE: Student should have knowledge of the algebraic, numerical, graphical and verbal representations of trigonometric functions, specifically the sine function. Students should have knowledge of logarithmic functions. Students should have knowledge of geometric sequences. Students should have basic knowledge of MATLAB programming, including writing functions and plotting graphs.

PROCEDURES: Use the *Teacher Notes* to guide the class lecture and discussion. Then students complete MATLAB based *Sine Sound Project.*

EVALUATION: *Sine Sound Project.*

REFERENCES:

[1] Zurasky, Jacob. *AEGIS RET Speech Processing Technical Report.* Florida Institute of Technology, July 2012.

[2] Stewart, James, and L. Redlin. *Precalculus: Mathematics for Calculus*. 5th ed. Australia: Brooks/Cole, 2010. p.444

[3] *MathWorks Product Documentation.* http://www.mathworks.com/help/techdoc/ref/soundsc.html

[4] *Piano Frequencies.* http://en.wikipedia.org/wiki/Piano_key_frequencies

[5] Këpuska, Veton. *Speech Sounds of American English.* http://my.fit.edu/~vkepuska/ece5526/Ch2-Speech_Sounds_of_US_English.pptx

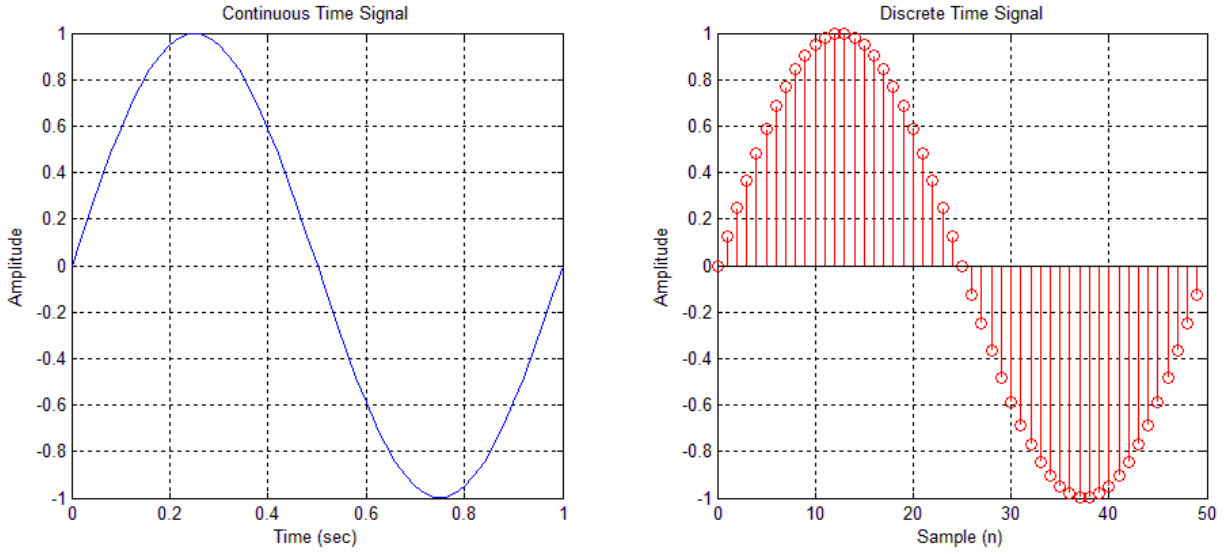
The Sound of a Sine Wave Teacher Notes

For additional information, please read *AEGIS Speech Processing Technical Report* [1].

1. Trigonometric functions are used to model periodic behavior, known as *harmonic motion*. The production of sound is an example of harmonic motion. Sound travels through the air as variation in air pressure. The frequency of the function determines the tone, and the amplitude determines the loudness [2]. A single sine wave represents pure tone.

2. A *continuous* sine wave is represented by the function: $y(t) = a \sin \omega t$.
   a. input*t* is time, output *y(t)* is amplitude at time *t*.
   b. amplitude = $|a|$ (maximum displacement), period $T = \frac{2\pi}{\omega}$ (length of time for one complete period or cycle), and there is no phase shift in this example.
   c. The *frequency* is the number of cycles per second, measured in Hertz (Hz). *frequency* $f = \frac{\omega}{2\pi}$ or $\omega = 2\pi f$. Notice $f = \frac{1}{T}$ (frequency is the reciprocal of period).
   d. So, substituting $\omega = 2\pi f$, the sine function written in terms of frequency is $y(t) = a \sin(2\pi f \cdot t)$

3. A computer uses a *discrete* representation of a sine wave.
   a. Sine wave is represented discretely by a sequence of samples taken over time.



b. *Sampling rate (fs)* is the number of samples per second. A *sampling rate* of 8000 Hz is typically used for telephone systems and speech signals. CD audio has a sampling rate of 44.1 kHz and DVD audio has a sampling rate of 96 kHz.

c. Period *T* is the length of time for one complete period (cycle) of the sine wave. In a discrete representation, the period is the number of samples for one complete period (cycle). $T = \frac{fs}{f} = \frac{samples/\sec}{periods/\sec} = \frac{samples}{period}$

d. Frequency $f = \frac{1}{T}$. So in a discrete representation, $f = \frac{f}{fs}$

e. So, substituting $f = \frac{f}{fs}$, a *discrete* representation of sine wave is:

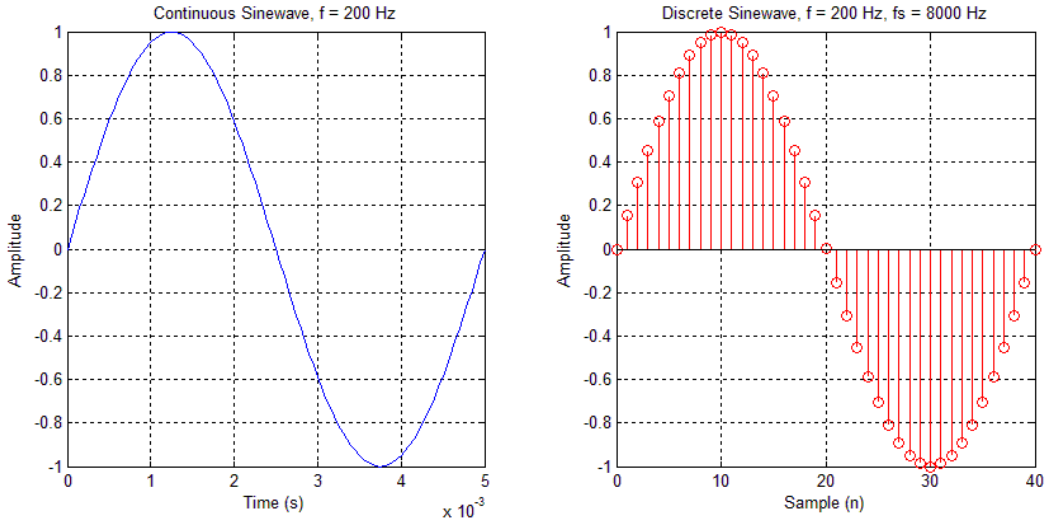
$y[n] = a\sin\left[2\pi \cdot \left(\frac{f}{fs}\right) \cdot n\right]$ where *n* is the sample number *n = 0,1,2,..*

4. Example.

a. Continuous sine wave $y = \sin(2\pi \cdot 200)t$ has frequency f=200 Hz and period $T = \frac{1}{200}$ seconds.

b. Given a sampling rate fs = 8000Hz, the discrete representation of this function is:

$y[n] = \sin\left[2\pi \cdot \left(\frac{200}{8000}\right) \cdot n\right]$.



5. A *composite signal* is a combination of several signals. A composite signal can be modeled mathematically by combining functions (adding, subtracting, etc.).

6. MATLAB command for playing audio.

    a. `soundsc(y, Fs)` sends audio signal *y* to the speaker at sample rate *Fs*. [3]

The Sound of a Sine Wave Student Notes

1. What is *harmonic motion*? What determines the tone and the volume of a sound? What type of function models pure tone?

2. A *continuous* sine wave is represented by the function: $y(t) = a \sin \omega t$.

   a. What is the input and the output of the function?

   b. Define the parameters:

      i. amplitude

      ii. period

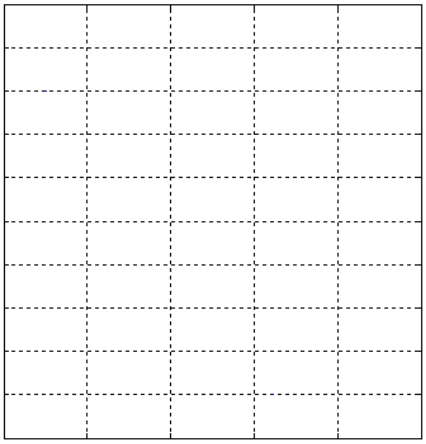
      iii. frequency

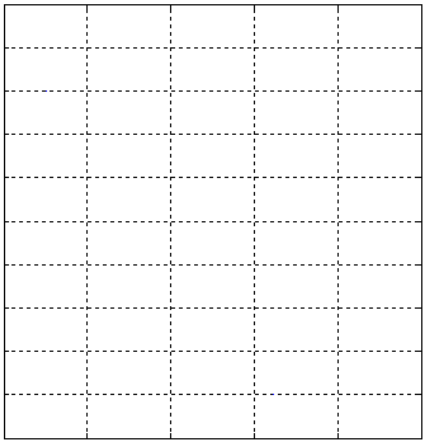
   c. Write the sine function in terms of frequency.

3. A computer uses a *discrete* representation of a sine wave

   a. What is the input of a discrete sine function?

   b. Sketch the graphs of continuous and discrete sine waves. Label the axes.





c. Define *sampling rate*. What sampling rates are typically used for speech signals, CD audio, and DVD audio?

d. Define the parameters for a discrete sine function:

   i. Period

   ii. Frequency

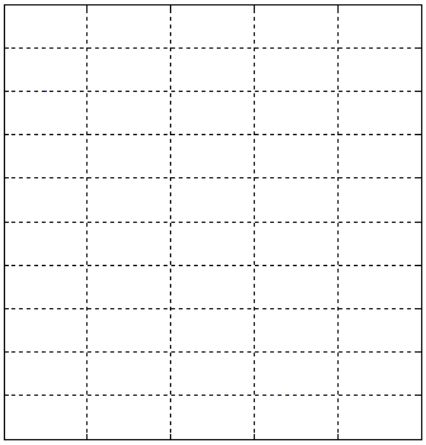
e. Write the equation for a *discrete* representation of sine wave in terms of frequency.

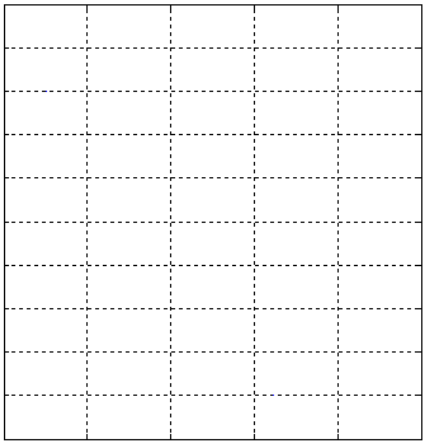
4. Example of continuous sine wave $y = \sin(2\pi \cdot 200)t$

   a. What is the frequency and the period?

   b. Write the discrete representation of this function, given a sampling rate *fs* = 8000 Hz.

   c. Sketch the graphs of the continuous and discrete representations of the function. Label the axes.





5. What is a *composite signal* and how is it modeled?

6. What MATLAB command is used for playing an audio signal? Define the inputs to the function.

Sine Sound Project

You are going to create three sine waves with different frequencies and a composite signal of the three sine waves. You will then display the graphs and play the sounds of the sine waves and the composite signal.

MATLAB Program

Write a MATLAB program to do the following. Your code must be commented.

1. Create `function sine_sound(fs, f1, f2, f3, time )`

   Inputs to the function:

   ```
   % fs   - sampling rate
   % f1   - first frequency
   % f2   - second frequency
   % f3   - third frequency
   % time - length of signals in seconds
   ```

2. Create three sine waves with frequencies `f1, f2, f3.` For each sine wave, use a sampling rate of `fs.` Generate `time` seconds worth of samples.

3. Create a composite sine wave of the three sine waves (add the sine waves together).

4. Graph the sine waves and the composite sine wave. Use the `subplot` command to create a 4x1 figure and plot all four graphs on the same figure. Label the axes and title each graph.

5. Use the `soundsc` command to play the sound of each sine wave.

6. Call your function with the following input arguments:

   - `fs   = 8000 Hz (samples per second)`
   - `f1   = 261.626 Hz`
   - `f2   = 523.251 Hz`
   - `f3   = 1046.500 Hz`
   - `time = 1 second`

Analysis

Compare the graphs and the sounds of the sine waves. Include a copy of your code and the figure output from your program. Consider the following in your analysis.

1. How does increasing the frequency affect the graph?

2. How does increasing the frequency affect the sound produced by each sine wave?

3. Choose a sample number, n. Using the data cursor, find the amplitude of each sine wave at sample number n. Verify the amplitude of the composite function at sample n is the sum of the output values of the other 3 functions at sample n.

4. What happens in the sound output of the composite signal?

5. What does the amplitude of the function represent in terms of the graph of the functions? In terms of the sound?

6. What musical notes do these three frequencies represent (do an Internet search on "piano frequencies")? What happens to frequency as you increase or decrease by an octave? What type of scale (linear or logarithmic) would best represent the frequencies of the notes of a piano and why?

7. The following formula gives the frequency f of the n^{th} key of a piano:
   $f(n) = 2^{\frac{n-49}{12}} \times 440Hz$.

   a. Use this formula to verify the frequency of the 1st key (A0) and the 88th key (C8) of a piano. What is the frequency range of a piano?

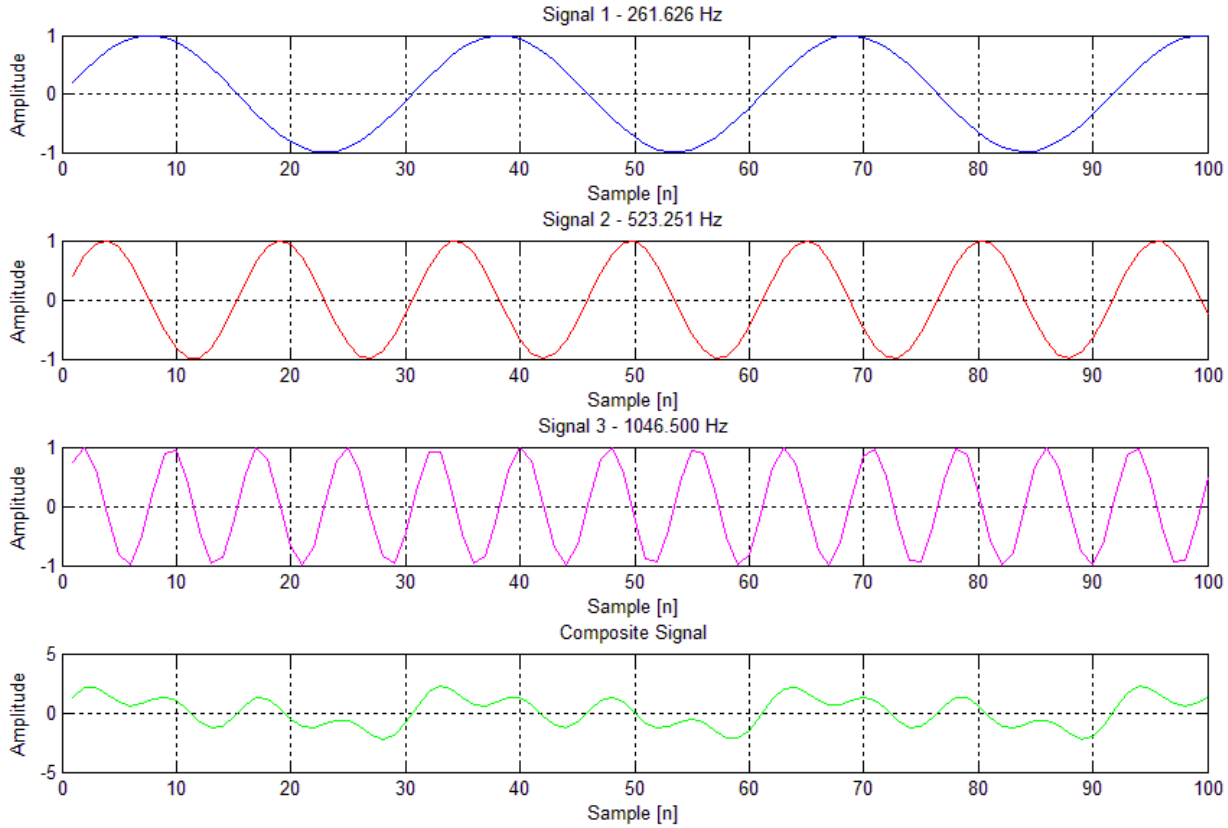
   b. Use this formula to verify the frequency of middle A4 (49th key) and A#4 (50th key). Identify and verify the common ratio between successive notes. What type of sequence is this?

Sine Sound Project – Sample Code

```
functionsine_sound_sample(fs,f1,f2,f3,time)
%
% plots graphs of three sine waves with frequencies f1, f2, f3
% plots graph of composite signal (addition of sine waves)
% play each sound wave and the composite
%
% fs   - sampling rate
% f1   - first frequency
% f2   - second frequency
% f3   - third frequency
% time - length of signals in seconds
%
% plays C4, C5, C6 – notice frequencies double between octave
% sine_sound_sample(8000, 261.626, 523.251, 1046.500, 1);
    %

closeall;

% number of samples in time period
n = 1:fs*time;

% generate signals
y1 = sin(2 * pi * (f1/fs) * n);
y2 = sin(2 * pi * (f2/fs) * n);
y3 = sin(2 * pi * (f3/fs) * n);

%composite signal
y4 = y1 + y2 + y3;

% plot sine waves
figure;
holdon;

subplot(4,1,1);
plot(n(1:100),y1(1:100), 'b');
gridon;
xlabel('Sample [n]');
ylabel('Magnitude');
title(sprintf('Signal 1 - %0.3f Hz', f1));

subplot(4,1,2);
plot(n(1:100), y2(1:100), 'r');
gridon;
xlabel('Sample [n]');
ylabel('Magnitude');
title(sprintf('Signal 2 - %0.3f Hz', f2));

subplot(4,1,3);
plot(n(1:100), y3(1:100), 'm');
gridon;
xlabel('Sample [n]');
ylabel('Magnitude');
title(sprintf('Signal 3 - %0.3f Hz', f3));
```

```
subplot(4,1,4);
plot(n(1:100), y4(1:100), 'g');
gridon;
xlabel('Sample [n]');
ylabel('Magnitude');
title('Composite Signal');

y = [y1 y2 y3 y4];

% play sine waves
soundsc(y, fs);

end
```

Sine Sound Project – Sample Output and Analysis



1. The higher the frequency, the shorter the period. The graph of the sine wave becomes more compressed horizontally as the frequency increases.

2. The higher the frequency, the more compressed the wave, the higher the pitch of the sound.

3. For n=10: amplitude of signal 1 = 0.8851, signal 2 = -0.8238, signal 3 = 0.934. For the composite signal, the amplitude at n=10 is 0.9954 because we added the functions together. (0.8851 + -0.8282 + 0.934 = 0.9954).

4. All the sounds are combined and played at the same time. The sound is composed of all three frequencies.

5. The amplitude represents the maximum height of the graph of the function. The amplitude also represents the volume of the sound (*this may not be obvious to students based on this project*).

6. 261.626 Hz is Middle C (C4), 523.251 Hz is Tenor C (C5), and 1046.500 Hz is Soprano C (C6). As you increase an octave, the frequency is doubled. A logarithmic scale would best represent the frequencies of piano notes because the frequencies are doubling every octave.

7. a. 1st key (A0): $f(1) = 2^{\frac{1-49}{12}} \times 440 = 2^{-48} \times 440 = 27.5Hz$;

   88th key (C8): $f(88) = 2^{\frac{88-49}{12}} \times 440 = 2^{\frac{39}{12}} \times 440 = 4186.01Hz$;

   $f(88) - f(1) = 4186.01 - 27.5 = 4158.51Hz$ difference between first and last key.

   b. 49th key (A4, Middle A): $f(49) = 2^{\frac{49-49}{12}} \times 440 = 1 \times 440 = 440Hz$;

   50th key (A#): $f(50) = 2^{\frac{50-49}{12}} \times 440 = 2^{\frac{1}{12}} \times 440 = 466.164Hz$;

   $f(50)/f(49) \approx 1.05946$ ratio between 49th and 50th key.

   The common ratio between notes is $\sqrt[12]{2} \approx 1.05946$, therefore the frequencies double every 12 notes (octave). This is a geometric sequence.

Sine Sound Project Extension – Music Notes

The Sine Sound Project can easily be extended to play simple songs.

In the Sine Sound Project, students created and played the sounds of sine waves with frequencies of Middle C, Tenor C, and Soprano C. Have students modify their program to play musical notes that form a song. Frequencies for piano notes can be found at http://en.wikipedia.org/wiki/Piano_key_frequencies [4]. Students can play "chords" by creating a composite sine wave, as they did in the Sine Sound Project. Duration of the note is based on the time of the sine wave in terms of the number of samples.

The following sample code plays the musical notes for *Twinkle Twinkle Little Star.*

Music Notes Sample Code

```
closeall;

%beats per minute
tempo = 200;

%sampling rate
fs    = 8000;

% number of samples in one beat (in seconds)
samples_perbeat = 60 / tempo * fs;

% twinkle twinkle little star notes and duration
% C4-Q C4-Q G4-Q G4-Q A4-Q A4-Q G4-H

% duration of notes
whole_note   = 4*samples_perbeat;
half_note    = 2*samples_perbeat;
quarter_note = samples_perbeat;
eight_note   = 0.5*samples_perbeat;

% frequency of notes
C4 = 262.626;
G4 = 391.995;
A4 = 440.000;

% rest for 1/16 note
rest = zeros(1,0.5*eight_note);

% output array to hold sequence of notes
output = [];

%C4 Quarter note
freq = C4;
n    = 1:quarter_note;
y = sin(2*pi*(freq/fs)*n);
output = [output y];
```

```
% rest
outpout = [output rest];

%another C4 Quarter note
output = [output y];

%G4 Quarter note
freq = G4;
n    = 1:quarter_note;
y = sin(2*pi*(freq/fs)*n);
output = [output y];

% rest
outpout = [output rest];

%another G4 Quarter note
output = [output y];

%A4 Quarter note
freq = A4;
n    = 1:quarter_note;
y = sin(2*pi*(freq/fs)*n);
output = [output y];

% rest
outpout = [output rest];

%another A4 Quarter note
output = [output y];

%G4 Half note
freq = G4;
n    = 1:half_note;
y = sin(2*pi*(freq/fs)*n);
output = [output y];

% play song
soundsc(output, fs);

% write to wave file
wavwrite(output, fs, 'twinkle.wav');
```

Sine Sound Music Notes Project

You are going to write a script file to play some simple songs using sine waves. Use your sine_sound code to help you. Frequencies for music notes can be found at http://en.wikipedia.org/wiki/Piano_key_frequencies.

You may find the following bits of code helpful:

- Define your music notes as variables and set them equal to the frequency of the note. Then you can reuse them to play different songs. For example:

```
C4  = 261.626;
```

- Duration of a note is based on the time of the sine wave in terms of the number of samples. The following defines tempo as 200 beats per minute. Change this value to suit your music.

```
% beats per minute
tempo = 200;

% sampling rate
fs    = 8000;

% number of samples in one beat (in seconds)
samples_perbeat = 60/tempo * fs;

% duration of notes
whole_note   = 4 * samples_perbeat;
half_note    = 2 * samples_perbeat;
quarter_note = samples_perbeat;
eight_note   = 0.5 * samples_perbeat;
```

- To "rest", create a matrix of zeros using the 'zeros' command. For example:

```
% rest for 1/16 note
rest = zeros(1,0.5*eight_note);
```

- The following sample code initializes an empty matrix called "output" to hold the notes. Then a C4 quarter note is added to the matrix, followed by a rest.

```
% initialize empty matrix to hold music notes
output = [];

%C4Q
freq = C4;
n = 1:quarter_note;
y = sin(2*pi*(freq/fs)*n);
output = [output y];

% rest
output = [output rest];
```

- You can play “chords” by creating a composite sine wave (add the notes together just like you did in sine_sound)

If you are a musician, you may write your own songs. Otherwise, below are a couple of measures from some simple songs.

The notes below use the following notation:

W = whole note
H = half note
Q = quarter note
E = eighth note
R = rest

```
% Twinkle Twinkle Little Star
% C4Q C4Q G4Q G4Q | A4Q A4Q G4H | F4Q F4Q E4Q E4Q | D4Q D4Q C4H

% Super Mario Bros.
% E4Q E4Q RQ E4Q | RQ C4Q E4Q RQ | G4Q

% Ode To Joy
% E4Q E4Q F4Q G4Q | G4Q F4Q E4Q D4Q | C4Q C4Q D4Q E4Q | E4Q D4Q D4H

% Frere Jacques
% C4Q D4Q E4Q C4Q | C4Q D4Q E4Q C4Q | E4Q F4Q G4H | E4Q F4Q G4H

% Jingle Bells
% E4Q E4Q E4Q RQ | E4Q E4Q E4Q RQ | E4Q G4Q C4Q D4Q | E4H

% Star Gazing
% C4H G4H | C4H G4H | F4Q E4Q D4Q E4Q | C4H RH
```

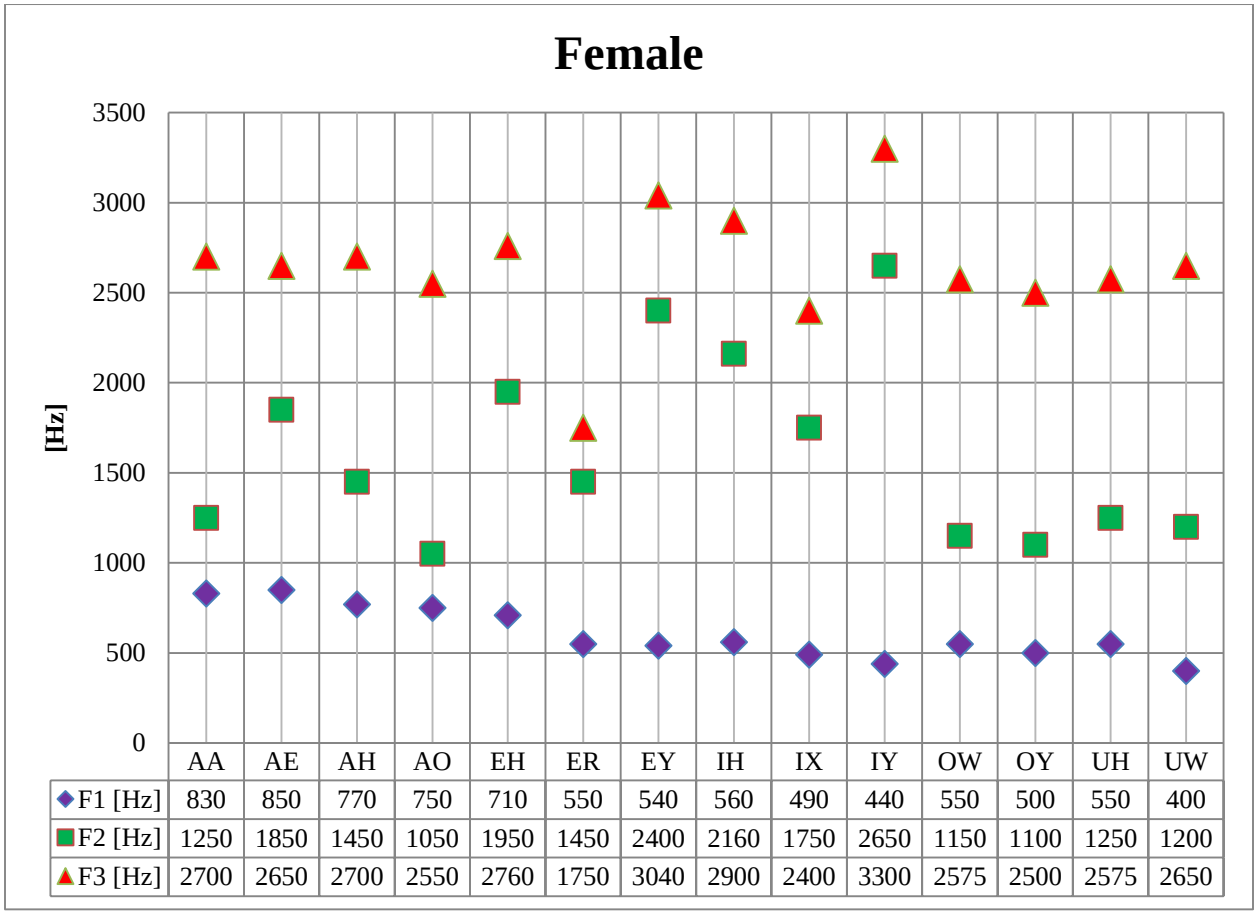
Sine Sound Project Extension – Vowel Sounds

The Sine Sound Project can easily be extended to synthesize vowel sounds.

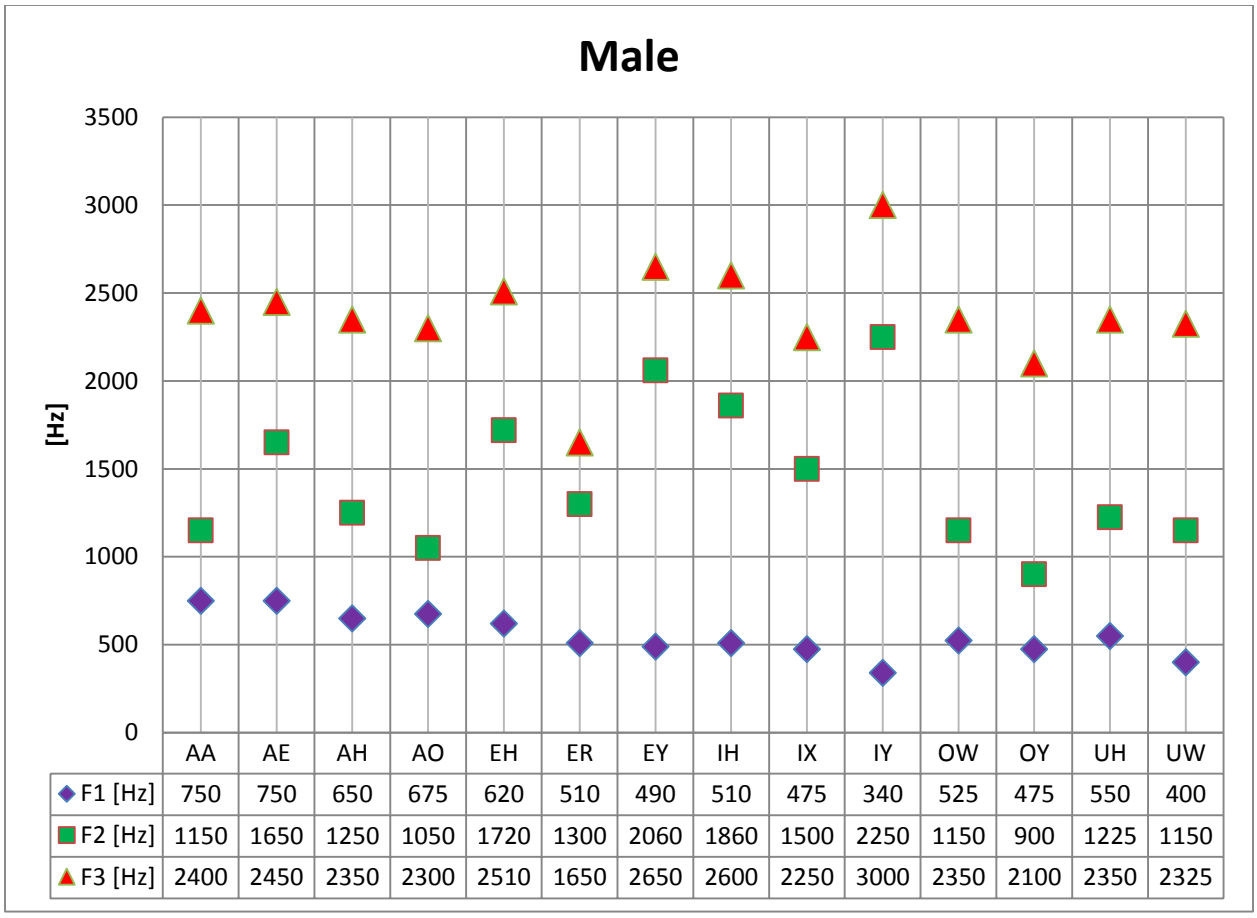
Vowels are characterized by the three lower formant frequencies. The following graphs give the average frequency and duration for vowel sounds by female and male speakers [5].

Vowel Sounds

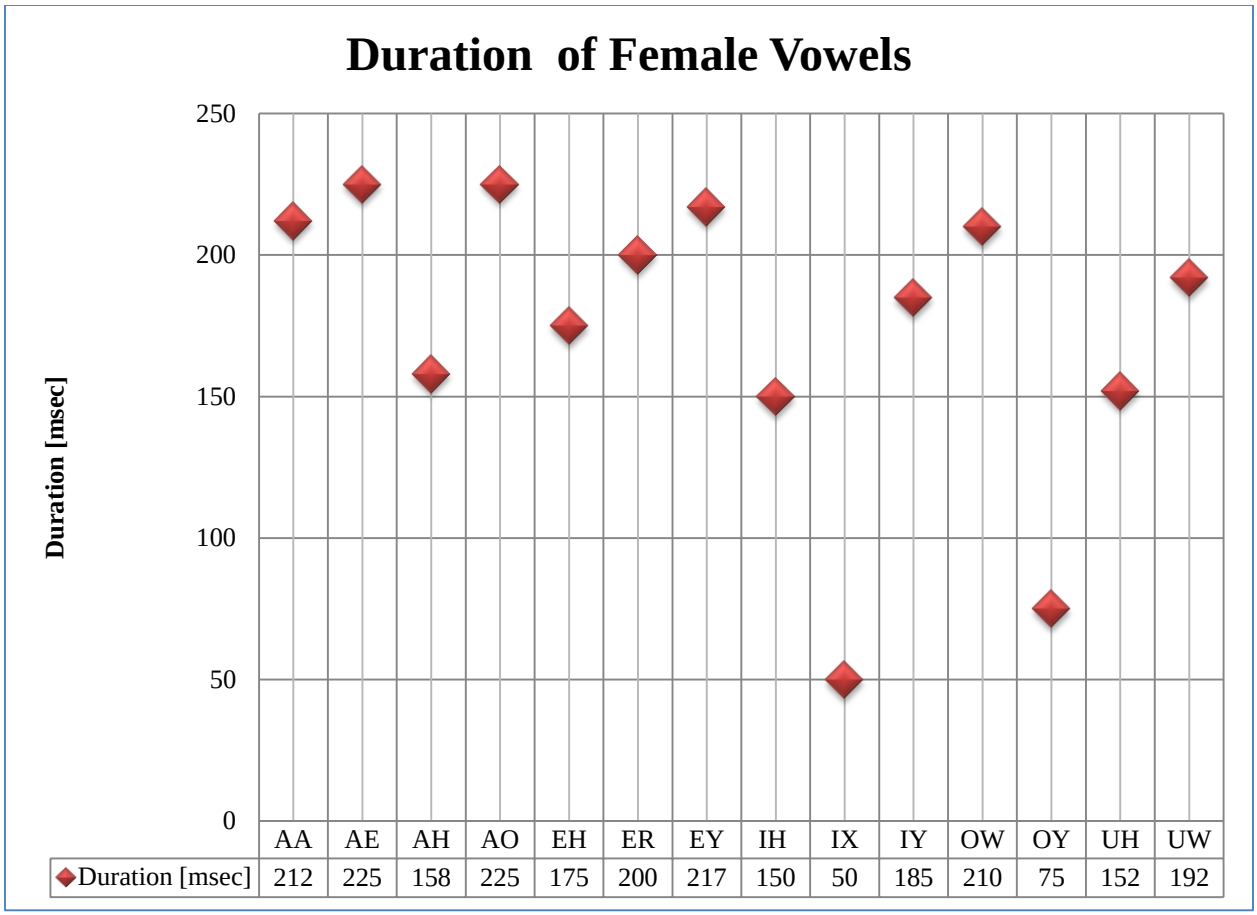
Symbol	Example	Symbol	Example	Symbol	Example
IY	B**ea**t	AO	B**ou**ght	AY	B**i**te
IH	B**i**t	AH	B**u**t	OY	B**oy**d
EY	B**ai**t	OW	B**oa**t	AW	B**ou**t
EH	B**e**t	UH	B**oo**k	AX	**A**bout
AE	B**a**t	UW	B**oo**t	IX	Ros**e**s
AA	B**o**b	ER	B**er**t	AXR	Butt**er**

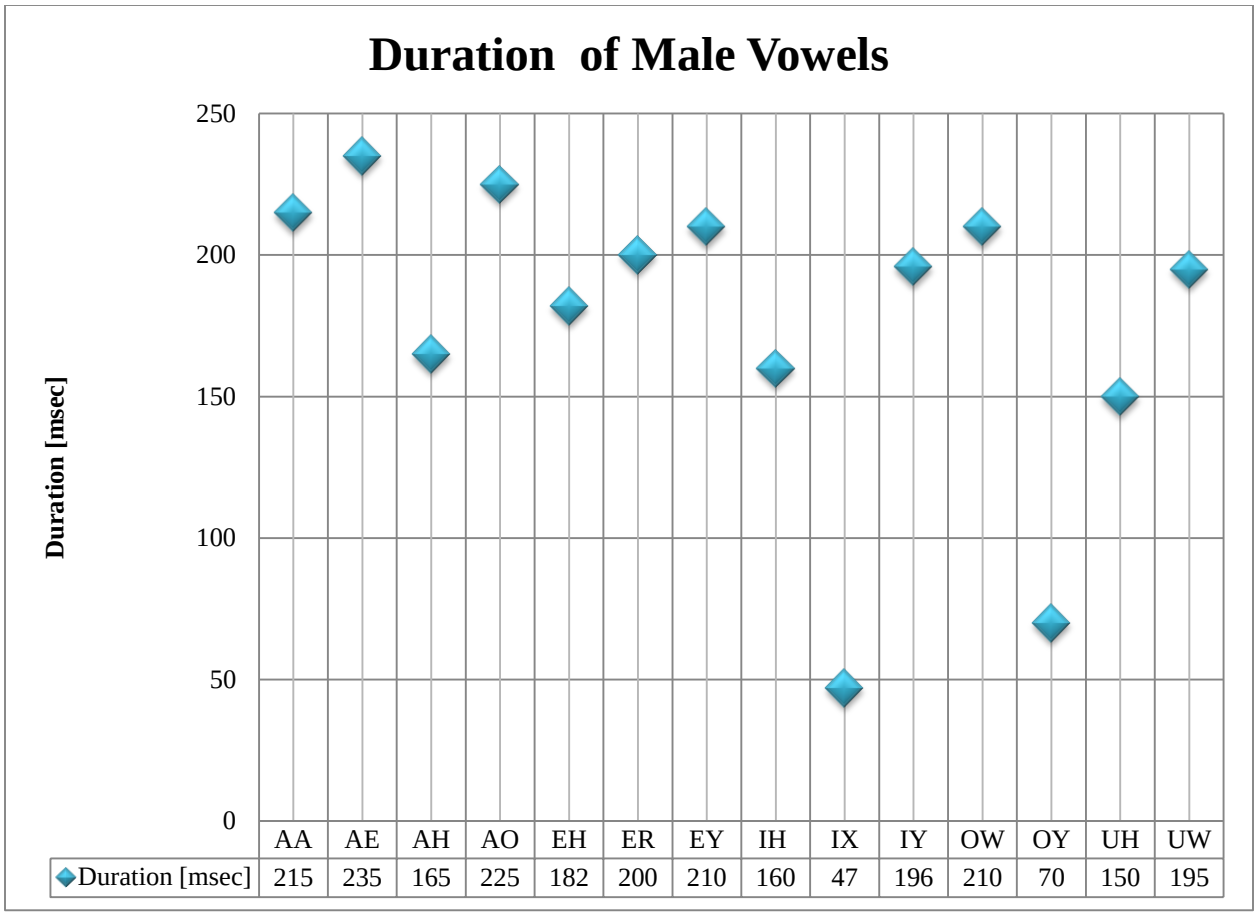


	AA	AE	AH	AO	EH	ER	EY	IH	IX	IY	OW	OY	UH	UW
F1 [Hz]	830	850	770	750	710	550	540	560	490	440	550	500	550	400
F2 [Hz]	1250	1850	1450	1050	1950	1450	2400	2160	1750	2650	1150	1100	1250	1200
F3 [Hz]	2700	2650	2700	2550	2760	1750	3040	2900	2400	3300	2575	2500	2575	2650



	AA	AE	AH	AO	EH	ER	EY	IH	IX	IY	OW	OY	UH	UW
F1 [Hz]	750	750	650	675	620	510	490	510	475	340	525	475	550	400
F2 [Hz]	1150	1650	1250	1050	1720	1300	2060	1860	1500	2250	1150	900	1225	1150
F3 [Hz]	2400	2450	2350	2300	2510	1650	2650	2600	2250	3000	2350	2100	2350	2325


Duration of Female Vowels
Duration [msec]
250
200
150
100
50
0
AA AE AH AO EH ER EY IH IX IY OW OY UH UW
Duration [msec] 212 225 158 225 175 200 217 150 50 185 210 75 152 192


Duration of Male Vowels
Duration [msec]
250
200
150
100
50
0
AA AE AH AO EH ER EY IH IX IY OW OY UH UW
Duration [msec] 215 235 165 225 182 200 210 160 47 196 210 70 150 195

Vowel Sounds Sample Code

Below is sample code to play a vowel sound. The function is called with the three frequencies that compose the vowel sounds and the duration.

For example, the AA vowel sound (as in “Bob”) for a male speaker is characterized by the frequencies 750, 1150, 2400 with a duration of 215 ms. Calling the sine_sound_vowels function with these input arguments will play the AA vowel sound.

```
% vowel sound aa, male "Bob"
sine_sound_vowels(8000, 750, 1150, 2400, 215);
```

The following function call plays the iy vowel sound (as in “beat”).

```
% vowel sound iy, male "beat"
sine_sound_vowels(8000, 340, 2250, 3000, 196);
```

It is interesting to play several vowel sounds and notice the difference.This is a rudimentary synthesis of a vowel sound, based only on three frequencies of equal weights. In human speech, vowel sounds consist of many more harmonic frequencies with different weights. Additionally, it can be difficult to recognize the sound when it is not in the context of a word.

The codebelow is very similar to the sine_sound_sample function from the *Sine Sound Project* sample code. The original function may be used, but notice the vowel durations are given in ms, so this value would need to be converted to seconds.

```
functionsine_sound_vowels(fs, f1, f2, f3, d)
%
% plots graphs of three sine waves with frequencies f1, f2, f3
% plots graph of composite signal (addition of sine waves)
% play each sound wave + composite
%
% fs   - sampling rate
% f1   - first frequency
% f2   - second frequency
% f3   - third freuency
% d    - duration of sound in ms
%
% vowel sound aa, male "Bob"
% sine_sound_vowels(8000, 750, 1150, 2400, 215);
%
% vowel sound iy, male  "beat"
% sine_sound_vowels(8000, 340, 2250, 3000, 196);
%
closeall;

% convert to seconds
time = d / 1000;
% double the duration
time = time * 2;
```

```
% number of samples in time period
    n = 1:fs*time;

% generate signals
    y1 = sin(2 * pi * (f1/fs) * n);
    y2 = sin(2 * pi * (f2/fs) * n);
    y3 = sin(2 * pi * (f3/fs) * n);

%composite signal
    y = y1 + y2 + y3;
% normalize
    y = y ./ max(abs(y));

% plot sine waves
figure;
holdon;

subplot(4,1,1);
plot(n(1:100),y1(1:100), 'b');
gridon;
xlabel('Sample [n]');
ylabel('Magnitude');
title(sprintf('Signal 1 - %0.3f Hz', f1));

subplot(4,1,2);
plot(n(1:100), y2(1:100), 'r');
gridon;
xlabel('Sample [n]');
ylabel('Magnitude');
title(sprintf('Signal 2 - %0.3f Hz', f2));

subplot(4,1,3);
plot(n(1:100), y3(1:100), 'm');
gridon;
xlabel('Sample [n]');
ylabel('Magnitude');
title(sprintf('Signal 3 - %0.3f Hz', f3));

subplot(4,1,4);
plot(n(1:100), y(1:100), 'g');
gridon;
xlabel('Sample [n]');
ylabel('Magnitude');
title('Composite Signal');

% play vowel sound
soundsc(y, fs);

% write to wave file
wavwrite(y, fs, 'vowel.wav');

end
```

Sound of a Sine Wave Report

Conclusion. In paragraph form, summarize the lessons you completed in the lab this week.

Reflection. Following your summary, reflect on your learning and your experiences during the Speech Processing lessons. Consider the following questions:

- What did you learn/gain from this project?

- Do you feel this project furthered your understanding of applications of trigonometric functions? How does this compare to studying applications of trig functions with word problems from the textbook?

- What was your "favorite" lesson or activity from this previous week and why?

- What did you think about learning and using MATLAB? Would you like to learn more about MATLAB? Do you have any ideas for how you would like to use MATLAB again in the future?

- Has this project influenced your interest or changed your mind about studying science, technology, engineering, and mathematics in the future?

- Has your viewpoint on the role of mathematics in the "real world" changed as a result of this project? Students often ask me "When am I ever gonna use this in real-life?" Do you think this project has helped to address the question of how math is used in the "real world"?

- Has the project given you a better understanding or appreciation of the technology you use in your everyday life?

- Do you have any constructive suggestions to me for improving these lessons for use in future classes?

Portfolio. Put together all of your work from this past week. Points may be added or deducted for neatness. Your portfolio must include the following items:

- Title page
- Intro to MATLAB worksheet
- 3 applications of Speech Processing homework assignment
- 'Sound of a Sine Wave' lecture notes
- 'Sound of a Sine Wave' project assignment sheet
- 'Sound of a Sine Wave' work - code, graph output, answer to analysis question
- Music Notes code or any additional work you did on the last day
- Conclusion and Reflection.

Lesson Plan 2: Frequency Analysis

NAMES: J. Rebecca Dowell and Jacob Zurasky

CONTENT AREA: Pre-Calculus

GRADE LEVEL: 9-12

TOPIC: Modeling Sound with Trigonometric Functions

NEXT GENERATION SUNSHINE STATE STANDARDS:

MA.912.A.2.2 Interpret a graph representing a real-world situation.

MA.912.A.2.7 Perform operations (addition, subtraction, division and multiplication) of functions algebraically, numerically, and graphically.

MA.912.A.2.13 Solve real-world problems involving relations and functions.

MA.912.D.11.2 Use sigma notation to describe series.

MA.912.T.1.6 Define and graph trigonometric functions using domain, range, intercepts, period, amplitude, phase shift, vertical shift, and asymptotes with and without the use of graphing technology.

MA.912.T.1.8 Solve real-world problems involving applications of trigonometric functions using graphing technology when appropriate.

MA.912.T.4.4 Define the trigonometric form of complex numbers, convert complex numbers to trigonometric form, and multiply complex numbers in trigonometric form.

COMMON CORE STATE STANDARD MATHEMATICS:

MACC.912.F-BF.1 Write a function that describes a relationship between two quantities.

MACC.912.F-BF.3 Identify the effect on the graph by replacing f(x) by f(x)+k, kf(x), f(kx), and f(x+k).

MACC.912.F-IF.4 For a function that models a relationship between two quantities, interpret key features of graphs and tables in terms of the quantities.

MACC.912.F-TF.5 Choose trigonometric functions to model periodic phenomena with specified amplitude, frequency, and midline.

MACC.912.N-CN.4 Represent complex numbers on the complex plane in rectangular and polar form (including real and imaginary numbers), and explain why the rectangular and polar forms of a given complex number represent the same number.

COMMON CORE MATHEMATICAL PRACTICE:
Model with mathematics.
Use appropriate tools strategically.

UNIT: Speech Processing

GOAL: Student will understand sound modeled as a composite of sinusoidal functions. Student will understand how to create harmonic signals. Student will understand the use of the Fourier Transform to transform a function of time into a function of frequency. Student will understand how to interpret a frequency spectrum graph.

OBJECTIVE: Student will model a harmonic signal with a series of sinusoidal functions. Student will denote the function using sigma notation. Using technology, student will graph and play the sound of the harmonic signal. Using technology, student will implement the Fourier Transform algorithm to transform the function from time domain to frequency domain. Using technology, student will graph the frequency spectrum of the harmonic signal and analyze the results.

MATERIALS: Graphing calculator. Computer with sound system (microphone and speakers).MATLAB software.

PRIOR KNOWLEDGE: Students should have knowledge of modeling sound with sinusoidal functions (prior completion of "The Sound of a Sine Wave" lesson is recommended). Student should have knowledge of trigonometric functions, specifically the sine function. Student should have knowledge of complex numbers (rectangular and polar form) and the complex plane. Students should have knowledge of series and sigma notation. Students should have basic knowledge of MATLAB programming, including writing functions, plotting graphs, and playing sounds.

PROCEDURES: Use the *Teacher Notes* to guide the class lecture and discussion. Then students complete MATLAB based *Frequency Analysis Project.*

EVALUATION: *Frequency Analysis Project.*

REFERENCES:
[1] Zurasky, Jacob. *AEGIS RET Speech Processing Technical Report.* Florida Institute of Technology, July 2012

[2] Weeks, Michael. *Digital signal processing using MATLAB and wavelets*. Hingham, Mass.: Infinity Science Press, 2007. p.140.

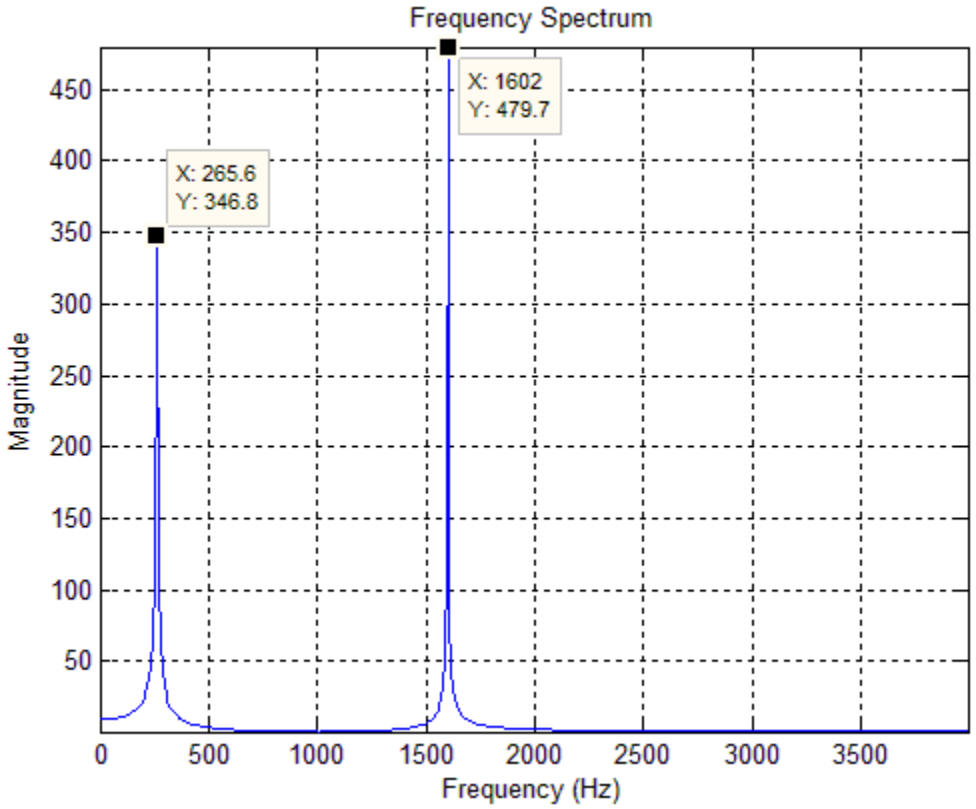
[3] *MathWorks Product Documentation.* http://www.mathworks.com/help/techdoc/ref/fft.html

Frequency Analysis Teacher Notes

For additional information, please read *AEGIS Speech Processing Technical Report* [1].

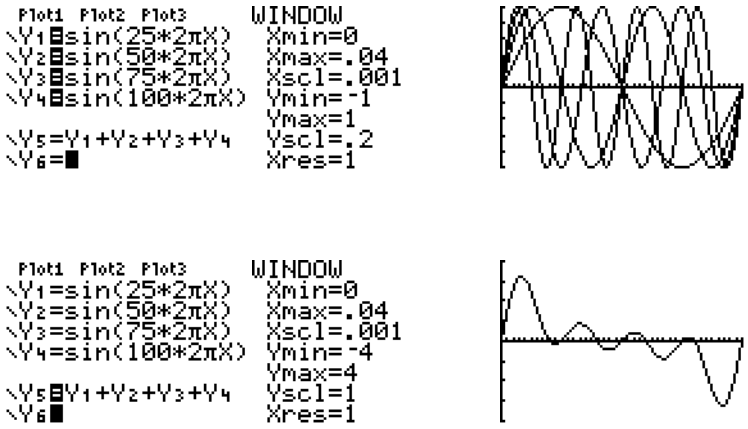
1. Sound signals can be modeled as an infinite combination of pure sine waves. The reverse is also true - a sound signal can be broken down, or decomposed into a series of sine waves that created that signal. Each sine wave component has a particular frequency and amplitude associated with it. The combination of these waves will reproduce the original sound signal.

   This process of *decomposition* is useful for knowing what frequencies are present in a sound signal and also the amplitude or strength of the signal at particular frequencies. The *Fourier Transform* is an algorithm that takes a discrete time signal as input and produces a frequency spectrum as output. The *frequency spectrum* is a graph of frequency vs. magnitude.



   In this example, the signal being analyzed is a composite signal of two sine waves with frequencies of 262.0 Hz and 1600.0 Hz. The Fourier transform was used on the signal to obtain the above frequency spectrum. The frequency spectrum shows a signal composed of two frequencies, 265.6 Hz and 1602.0 Hz. The signals have similar magnitudes.

2. A *harmonic signal* is a composite signal of a series of sinusoidal functions where each function has a frequency that is an integer multiple of a fundamental frequency [2]. The following example graphs a harmonic signal with a fundamental frequency of 25 Hz.



In the student project, an example of a harmonic signal, a *square wave*, is created from a sum of harmonic sine waves. A square wave is periodic but not sinusoidal. Square waves are typically found in digital circuits, for example a computer clock signal.

3. Analyzing the frequency spectrum of a sound tells us about the sound. Most speech occurs below 10 kHz. From the project in the previous lesson, the range of the piano is 27.5 Hz to 4186 Hz. Bells and cymbals have a frequency range of 16-32 kHz. Rhythm frequencies (e.g. bass notes) have a range 32 – 512 Hz.

4. The Fourier Transform is used to transform a discrete time signal to a function of discrete frequency. The output of the Fourier Transform is a sequence of magnitudes and phase angles represented as complex numbers, for a given frequency.

    a. For continuous time, Fourier Transform is defined as

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi} dx$$, where

x – time

ξ - frequency (Hertz).

b. For discrete time, the Discrete Fourier Transform is defined as

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-i 2\pi \frac{k}{N} n}$$, where

X – frequency spectrum

k – frequency bin

N – FFT size

n – sample number

$x[n]$ – input signal

c. The Fast Fourier Transform (FFT) is a more efficient implementation of the Discrete Fourier Transform (DFT). MATLAB uses FFT.

d. The Fourier Transform returns results as complex numbers. The complex numbers represent the magnitude and phase angle at a given frequency. MATLAB returns the complex results in rectangular form ($a+bi$). When analyzing the frequency spectrum, calculate the magnitude of the frequency as the absolute value of the complex number. $|z| = \sqrt{a^2 + b^2}$

e. The FFT may be computed using Euler's formula to represent the complex number, as opposed to the exponential form. Euler's formula: $e^{ix} = \cos x + i \sin x$.

f. Because the input discrete time signal contains only real numbers, and due to the periodic nature of the complex number component of the Fourier Transform, the FFT output will have symmetry at the halfway point of the x-axis. Therefore, only the first half of the FFT results is needed.

g. Each frequency bin k represents a *frequency bandwidth* (range of frequencies). To calculate the *frequency bandwidth* of each bin, divide the sampling rate *fs* by the FFT size. To calculate the frequency of each bin, multiply the bin number k by the frequency bandwidth. A greater FFT size gives you greater frequency resolution (narrower frequency bandwidth per bin).

5. Example of FFT computed by hand. FFT size N = 4 (four points on complex unit circle).

$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-i 2\pi \frac{k}{N} n}$	Discrete Fourier Transform
$X[k] = \sum_{n=0}^{N-1} x[n] \cdot \left[\cos\left(-2\pi \frac{k}{N} n \right) + i \sin\left(-2\pi \frac{k}{N} n \right) \right]$	DFT using Euler's Formula
$x[n] = [6,4,9,1]$	Input signal
$X[0] = 6 \cdot \left[\cos\left(-2\pi \cdot \frac{0}{4} \cdot 0 \right) + i \sin\left(-2\pi \cdot \frac{0}{4} \cdot 0 \right) \right] +$ $4 \cdot \left[\cos\left(-2\pi \cdot \frac{0}{4} \cdot 1 \right) + i \sin\left(-2\pi \cdot \frac{0}{4} \cdot 1 \right) \right] +$ $9 \cdot \left[\cos\left(-2\pi \cdot \frac{0}{4} \cdot 2 \right) + i \sin\left(-2\pi \cdot \frac{0}{4} \cdot 2 \right) \right] +$ $1 \cdot \left[\cos\left(-2\pi \cdot \frac{0}{4} \cdot 3 \right) + i \sin\left(-2\pi \cdot \frac{0}{4} \cdot 3 \right) \right]$ $= 6[1+0i] + 4[1+0i] + 9[1+0i] + 1[1+0i]$ $= 6 + 4 + 9 + 1$ $= 20$	Compute the DFT for N=4
$X[1] = 6 \cdot \left[\cos\left(-2\pi \cdot \frac{1}{4} \cdot 0 \right) + i \sin\left(-2\pi \cdot \frac{1}{4} \cdot 0 \right) \right] +$ $4 \cdot \left[\cos\left(-2\pi \cdot \frac{1}{4} \cdot 1 \right) + i \sin\left(-2\pi \cdot \frac{1}{4} \cdot 1 \right) \right] +$ $9 \cdot \left[\cos\left(-2\pi \cdot \frac{1}{4} \cdot 2 \right) + i \sin\left(-2\pi \cdot \frac{1}{4} \cdot 2 \right) \right] +$ $1 \cdot \left[\cos\left(-2\pi \cdot \frac{1}{4} \cdot 3 \right) + i \sin\left(-2\pi \cdot \frac{1}{4} \cdot 3 \right) \right]$ $= 6[1+0i] + 4[0-1i] + 9[-1+0i] + 1[0+1i]$ $= 6 - 4i - 9 + 1i$ $= -3 - 3i$	

$X[2] = 6 \cdot \left[\cos\left(-2\pi \cdot \frac{2}{4} \cdot 0 \right) + i \sin\left(-2\pi \cdot \frac{2}{4} \cdot 0 \right) \right] +$ $4 \cdot \left[\cos\left(-2\pi \cdot \frac{2}{4} \cdot 1 \right) + i \sin\left(-2\pi \cdot \frac{2}{4} \cdot 1 \right) \right] +$ $9 \cdot \left[\cos\left(-2\pi \cdot \frac{2}{4} \cdot 2 \right) + i \sin\left(-2\pi \cdot \frac{2}{4} \cdot 2 \right) \right] +$ $1 \cdot \left[\cos\left(-2\pi \cdot \frac{2}{4} \cdot 3 \right) + i \sin\left(-2\pi \cdot \frac{2}{4} \cdot 3 \right) \right]$ $= 6[1 + 0i] + 4[-1 + 0i] + 9[1 + 0i] + 1[-1 + 0i]$ $= 6 - 4 + 9 + -1$ $= 10$	
$X[3] = 6 \cdot \left[\cos\left(-2\pi \cdot \frac{3}{4} \cdot 0 \right) + i \sin\left(-2\pi \cdot \frac{3}{4} \cdot 0 \right) \right] +$ $4 \cdot \left[\cos\left(-2\pi \cdot \frac{3}{4} \cdot 1 \right) + i \sin\left(-2\pi \cdot \frac{3}{4} \cdot 1 \right) \right] +$ $9 \cdot \left[\cos\left(-2\pi \cdot \frac{3}{4} \cdot 2 \right) + i \sin\left(-2\pi \cdot \frac{3}{4} \cdot 2 \right) \right] +$ $1 \cdot \left[\cos\left(-2\pi \cdot \frac{3}{4} \cdot 3 \right) + i \sin\left(-2\pi \cdot \frac{3}{4} \cdot 3 \right) \right]$ $= 6[1 + 0i] + 4[0 + 1i] + 9[-1 + 0i] + 1[0 - 1i]$ $= 6 + 4i - 9 - 1i$ $= -3 + 3i$	
$\|X[0]\| = \sqrt{20^2 + 0^2} = 20$ $\|X[1]\| = \sqrt{(-3)^2 + (-3)^2} = \sqrt{18} \approx 4.2426$ $\|X[2]\| = \sqrt{10^2 + 0^2} = 10$ $\|X[3]\| = \sqrt{(-3)^2 + (3)^2} = \sqrt{18} \approx 4.2426$	Compute the magnitudes
The bandwidth of each bin depends on the sampling rate of the original signal. However, the results indicate the magnitude of the signal energy at frequency bin *X[0]* is 20, at *X[1]* is 4.2426, at *X[2]* is 10, and at *X[3]* is 4.2426. The *X[0]*bin is overall energy of the signal. Notice the symmetry between *X[1]* and *X[3]*.	

6. MATLAB commands to compute the FFT and display the resulting frequency spectrum

    a. `Y = fft(X,n)` – computes and returns *n*-point Fast Fourier Transform of *X* [3].

    b. Use the `stem` command to plot the results of the FFT. Because of the symmetry of the FFT results, plot the first half of the FFT. To make the x-axis graph labels meaningful, convert the array index of the FFT results to the corresponding frequency bands. Since FFT returns complex results, take the absolute value (`abs`) of the FFT results to obtain the magnitude of the frequencies.

    c. Sample code (also given in student project).

```
% number of frequency bands
freq_bands = 0:(fft_size / 2)-1;

% calculate the frequency for each band
freq_bands = freq_bands * ( (fs/2) / (fft_size/2) );

% plot the frequency spectrum
stem(freq_bands, abs( fft_results(1:(fft_size/2)) ));
```

Frequency Analysis Project

You are going to create a sine wave with a fundamental frequency, four harmonic sine waves based on the fundamental frequency, and a composite of the five sine waves. You will plot the graphs and play the sounds of the sine waves and the composite signal. You will then compute the FFT of the composite signal and display and analyze the frequency spectrum.

MATLAB Program

Write a MATLAB program to do the following. Your code must be commented.

1. Create `function square_wave(fs, f1, time, fft_size)`

   Inputs to the function:

   ```
   % fs       - sampling rate
   % f1       - fundamental frequency
   % time     - length of the signals in seconds
   % fft_size - size of fft
   ```

2. Create a sine wave, $y = \sin(2\pi f \cdot t)$, with a fundamental frequency f (input argument `f1`). Remember you need to write a discrete representation of the sine wave. Use a sampling rate of `fs` and create `time` seconds worth of samples.

3. Create four harmonic sine waves based on the fundamental frequency:
   $y = \frac{1}{3}\sin(3 \cdot 2\pi f \cdot t)$, $y = \frac{1}{5}\sin(5 \cdot 2\pi f \cdot t)$, $y = \frac{1}{7}\sin(7 \cdot 2\pi f \cdot t)$, $y = \frac{1}{9}\sin(9 \cdot 2\pi f \cdot t)$

4. Create a composite function of the five harmonic sine waves (add them together).

5. Graph the five sine waves and the composite signal. Use the `subplot` command to create a 3x1 figure. For the first plot, use the `hold` command to graph all five individual sine functions on the same plot, using a different color for each. Use the `legend` command to indicate the frequency of each sine wave. For the second plot, plot the composite signal.

6. Play the sound of each sine wave and the composite signal, using the `soundsc` command.

7. Compute the FFT of the composite signal, using the `fft`command with an FFT size of `fft_size`.

8. For the third plot, plot the frequency spectrum generated by the FFT, using the `stem` command. Label your x-axis with the frequency bands (instead of the array index). Sample code for computing the frequency bands and plotting the frequency spectrum:

```
% number of frequency bands
freq_bands = 0:(fft_size / 2)-1;

% calculate the frequency for each band
freq_bands = freq_bands * ( (fs/2) / (fft_size/2) );

% plot the frequency spectrum
stem(freq_bands, abs( fft_results(1:(fft_size/2)) ));
```

9. Call your function with the following input arguments:
   - `fs       = 8000 Hz (samples per second)`
   - `f1       = 262 Hz`
   - `time     = 1 second`
   - `fft_size = 256`

Analysis

Include the figure output from you program. Consider the following in your analysis.

1. Represent the composite signal (the sum of the five harmonic sine waves) in sigma notation.

2. The composite signal is the sum of k sine waves for $k = 1....5$. How you think the graph of the composite signal will change as $k \to \infty$? Verify your prediction by modifying your MATLAB program to increase the number of harmonic sine waves in the composite signal (try $k = 1...10$).

3. Using the data cursor, compare the frequency peaks of the frequency spectrum to the original frequencies of the individual sine waves. What do these frequency peaks represent? How do you think the frequency spectrum will change if you increase or decrease the FFT size? Verify your prediction by decreasing your FFT size to 128 and then increasing your FFT size to 1024 and record your results in a table.

4. What does the height of each frequency peak indicate?

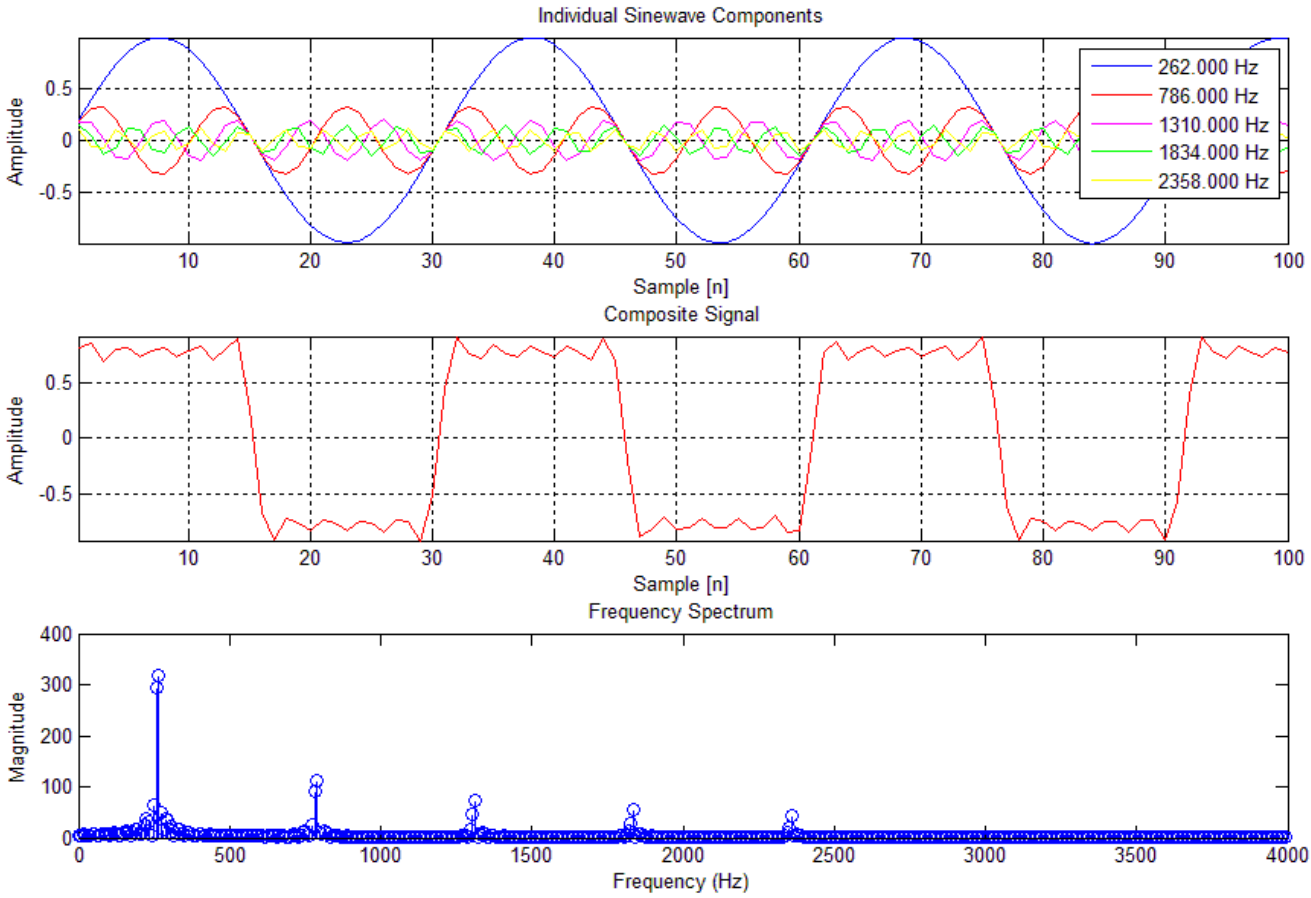
5. Why is it necessary to take the absolute value of the frequency spectrum returned by the FFT?

Frequency Analysis Project – Sample Code

```
functionsquare_wave(fs, f1, time, fft_size)
%
% create five harmonic signals based on the input fundamental frequency f1
% create composite signal
% compute fft
% plot graphs and play sounds
%
% fs       - sampling rate
% f1       - fundamental frequency
% time     - length of the signals in seconds
% fft_size - size of fft
%
% square_wave(8000, 262, 1, 256)
%

closeall;

%harmonic frequencies based on fundamental frequency
f2 = 3 * f1;
f3 = 5 * f1;
f4 = 7 * f1;
f5 = 9 * f1;

% number of samples in the time period
n = 1:fs * time;

% frequency band labels for frequency spectrum graph
freq_bands = 0:(fft_size / 2)-1;

%generate signals
y1 = sin(2 * pi * (f1 / fs) * n);
y2 = (1/3) * sin(2 * pi * (f2 / fs) * n);
y3 = (1/5) * sin(2 * pi * (f3 / fs) * n);
y4 = (1/7) * sin(2 * pi * (f4 / fs) * n);
y5 = (1/9) * sin(2 * pi * (f5 / fs) * n);

%composite signal
y = y1 + y2 + y3 + y4 + y5;

% play 3 individual tones, then the composite signal
soundsc(y1, fs);
soundsc(y2, fs);
soundsc(y3, fs);
soundsc(y4, fs);
soundsc(y5, fs);
soundsc(y, fs);

% compute FFT and frequency band labels
w = fft(y, fft_size);
freq_bands = freq_bands * ( (fs/2) / (fft_size/2) );
```

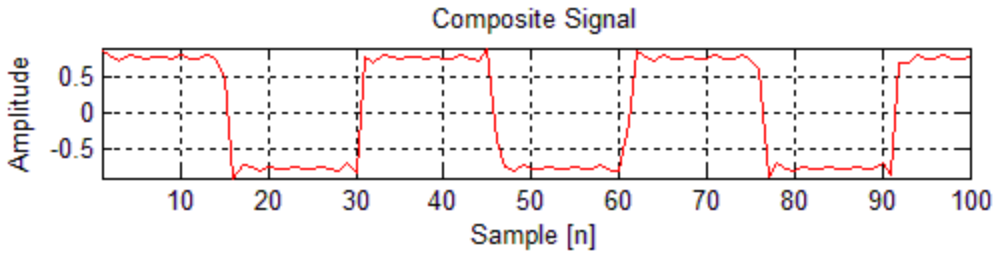
```
% plot all signals on one figure
figure;
subplot(3,1,1);
plot(n(1:100), y1(1:100), 'b');
holdon;
plot(n(1:100), y2(1:100), 'r');
holdon;
plot(n(1:100), y3(1:100), 'm');
holdon;
plot(n(1:100), y4(1:100), 'g');
holdon;
plot(n(1:100), y5(1:100), 'y');
holdon; grid on; axis tight;
xlabel('Sample [n]');
ylabel('Amplitude');
title('Individual Sinewave Components');
legend1 = legend(sprintf('%0.3f Hz', f1), sprintf('%0.3f Hz', f2),
sprintf('%0.3f Hz', f3), sprintf('%0.3f Hz', f4), sprintf('%0.3f Hz', f5));
set(legend1, 'Location', 'NorthEast');

% plot the composite signal
subplot(3,1,2);
plot(n(1:100), y(1:100), 'r');
axistight; grid on;
xlabel('Sample [n]');
ylabel('Amplitude');
title('Composite Signal');

% plot the frequency spectrum
subplot(3,1,3);
stem(freq_bands, abs( w(1:(fft_size/2)) ));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Frequency Spectrum');
```

Frequency Analysis Project - Sample Output and Analysis



1. The composite signal (sum of the five harmonic sine waves) represented in sigma notation: $y = \sum_{k=1}^{5} \frac{1}{2k-1} \sin((2k-1) \cdot 2\pi f \cdot t)$

2. As *k* increases, the composite signal becomes more "square". The top becomes flatter and the edges become steeper. Example output for *k =1…10*:



3. A higher fft size returns frequencies that are closer to the actual frequencies of the composite signal. A higher fft size results in higher frequency resolution. fft returns more results with a higher fft size (more data in the plot of the frequency spectrum).

Original Frequency	**FFT size**		
	128	**256**	**1024**
262.0	250.0	250.0	265.625
786.0	812.5	781.25	789.0625
1310.0	1312.5	1312.5	1312.5
1834.0	1812.5	1842.75	1835.9375
2358.0	2375	2343.75	2359.375

4. The height of the frequency spectrum indicates the magnitude or strength of the signal at that frequency. The greater magnitude indicates more of the signal energy is at that frequency.

5. We take the absolute value of the frequency spectrum because fft returns complex number results. A complex number can be represented in trig form in terms of magnitude and phase angle. For our frequency spectrum, we are interested in the magnitude of the frequencies.

Lesson Plan 3: Digital Sound Effects

NAMES: J. Rebecca Dowell and Jacob Zurasky

CONTENT AREA: Pre-Calculus

GRADE LEVEL: 9-12

TOPIC: Modeling Sound with Functions

NEXT GENERATION SUNSHINE STATE STANDARDS:

MA.912.A.2.2 Interpret a graph representing a real-world situation.

MA.912.A.2.7 Perform operations (addition, subtraction, division and multiplication) of functions algebraically, numerically, and graphically.

MA.912.A.2.13 Solve real-world problems involving relations and functions.

MA.912.D.1.2 Use finite differences to solve problems and to find explicit formulas for recurrence relations.

COMMON CORE STATE STANDARD MATHEMATICS:

MACC.912.N-Q.2 Define appropriate quantities for the purpose of descriptive modeling.

MACC.912.F-IF.3 Recognize that sequences are functions, sometimes defined recursively, whose domain is a subset of the integers.

MACC.912.F-IF.4 For a function that models a relationship between two quantities, interpret key features of graphs and tables in terms of the quantities.

MACC.912.F-BF.1 Write a function that describes a relationship between two quantities.

MACC.912.F-BF.3 Identify the effect on the graph by replacing f(x) by f(x)+k, kf(x), f(kx), and f(x+k).

COMMON CORE MATHEMATICAL PRACTICE:
Model with mathematics.
Use appropriate tools strategically.

UNIT: Speech Processing

GOAL: Student will understand modeling time-delay digital sound effects.

OBJECTIVE: Student will write a discrete function to model an echo sound effect using function operations and a delay function. Using technology, student will graph and play the sound of the function. Student will determine how changes in parameters affect the graph and the sound of the function.

MATERIALS: Computer with sound system (microphone and speakers). MATLAB software.Sound recording software that will create wave files, such as Audacity (available for free download at http://audacity.sourceforge.net/) [1].

PRIOR KNOWLEDGE: Students should have knowledge of discrete functions and function operations. Students should have basic knowledge of MATLAB programming, including writing functions, plotting graphs, and playing sounds.

PROCEDURES: Use the *Teacher Notes* to guide the class lecture and discussion. Then students complete MATLAB based *Digital Sound Effects Project.*

EVALUATION: *Digital Sound Effects Project.*

REFERENCES:

[1] *Audacity.* http://audacity.sourceforge.net/

[2] Zurasky, Jacob. *AEGIS RET Speech Processing Technical Report.* Florida Institute of Technology, July 2012.

[3] Këpuska, Veton. *Audio Processing Problems and Solutions.* http://my.fit.edu/~vkepuska/ece3551/Ch10-Audio Processing Problems and Solutions.pptx

[4] *Digital Audio Effects.* http://www.cs.cf.ac.uk/Dave/CM0268/PDF/10_CM0268_Audio_FX.pdf

[5] *Speed of Sound.* http://en.wikipedia.org/wiki/Speed_of_sound

[6] *MathWorks Product Documentation.* http://www.mathworks.com/help/techdoc/ref/wavread.html

[7] *MathWorks Product Documentation.* http://www.mathworks.com/help/techdoc/ref/wavwrite.html

[8] *MathWorks Product Documentation.* http://www.mathworks.com/help/techdoc/ref/fileparts.html

[9] *MathWorks Product Documentation.* http://www.mathworks.com/help/techdoc/ref/strcat.html

[10] *Absorption Coefficient Chart.*
http://www.sae.edu/reference_material/pages/Coefficient Chart.htm

Digital Sound Effects Teacher Notes

For additional information, please read *AEGIS Speech Processing Technical Report* [2].

1. Many sound effects use a time delayed version of the original sound. To create time delay sound effects digitally, a history of the original signal must be stored by the computer program. Types of time delay sound effects include:

    a. Echo. Simulates the reflection of sound off a surface. Time and volume of the echo depend on the distance to the surface and the reflection properties of the surface materials. Implemented by combining the original signal with a time delayed version of the original signal. The typical time delay is > 50 ms.

    b. Reverberation. Simulates many echoes of a sound. Implemented by combining the original signal with multiple delayed versions of the original signal.

    c. Chorus. Simulates multiple instruments or voices at the same time, each slightly varying time, pitch, and volume. Implemented by combining the original signal with a random time delay function. Typical delay time is 10-25 ms.

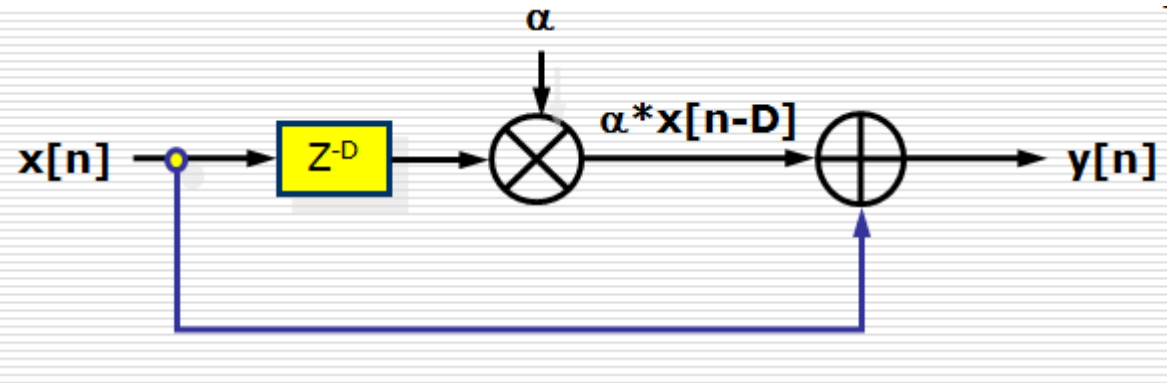
    d. Flanger. Simulates a "swooshing" jet sound, originally produced by mixing two tape decks varied with a small change in speed. Implemented by combining the original signal with a low frequency sinusoidal time delay function. Typical delay time 0-15 ms and sinusoidal modulation about 1 Hz.

    e. Vibrato. Simulates an oscillation in pitch, for example, the effect of a singer sustaining a note, a musician bending a stringed instrument, or a guitarist using a 'whammy' bar. Implemented by delaying the original signal using a low frequency sinusoidal delay function, which causes the pitch to vary. Typical delay time is 5-10 ms and time-delay frequency between 5-14 Hz.

    f. Tremolo. Simulates an oscillation in amplitude. Implemented by multiplying the original signal with a low frequency sinusoidal function, which causes the amplitude to vary. Typical frequency of the sinusoidal function is < 20 Hz.

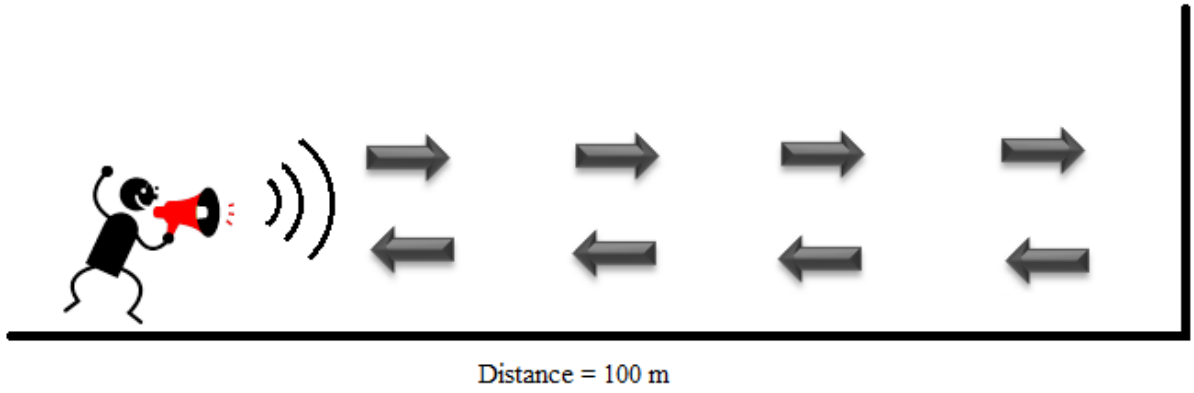
    [3] [4]

2. The echo effect can be represented by the signal flow graph shown below. The original signal $x[n]$ is added to a constant time delay function $x[n-D]$ multiplied by a constant alpha α. The output $y[n] = x[n] + \alpha * x[n-D]$



D represents the length of the delay in the amount in samples. The amount of the delay depends on the distance to the reflection surface. The further the distance to the surface, the greater the delay.

For example, assume the source of the sound (speaker) is located 100 m from a reflection surface (wall). The total distance *d* round trip from the speaker to the wall and back is 200 m. Assuming the speaker is at sea level in dry air at 20˚ C, the speed of sound $c_s$is 343.2 meters per second [5].



The time delay of echo t_e(sec) is calculated by dividing the total distance the sound travels *d* by the speed of sound c_s.

$$t_e = \frac{d}{c_s} = \frac{200m}{343.2m/\sec} \approx 0.588\,sec$$

Since we are working with discrete time signals, the time delay of the echo must be converted to delay in number of samples d_n. Assuming a sampling rate $f_s = 44.1kHz$,

Time delay of echo d_n(# samples) is calculated by multiplying the time delay of the echo by the sampling rate f_s.

$$d_n = t_e \cdot f_s = (0.588\sec)(44{,}100\,samples/\sec) = 25{,}941\,samples$$

So, the resulting echo function is

$$y[n] = x[n] + \alpha \cdot x[n - 25{,}941]$$

Alpha α represents the reflection coefficient of the reflection surface. The reflection coefficient is a ratio of the amount of the sound that is reflected off the surface. Different materials have different reflection properties. For example, heavy draperies that have a reflection coefficient of .25 reflect 25% of the sound. Concrete that has a reflection coefficient of .94 reflects 94% of the sound, resulting in a louder echo.

3. MATLAB commands to read and write wave audio files, used in Student Project.

   a. `[x, fs] = wavread(filename)` – loads a wave file specified by *filename,* returns the sampled data in *x* and the sampling rate *fs.* [6]

   b. `wavwrite(y, fs, filename)` – writes the data stored in *y* to a wave file *filename.* The data has a sample rate of *fs.* [7]

   c. `[path, name, ext] = fileparts(filename)` - returns the pathname, filename, and extension of the specified file. [8]

   d. `combinedStr = strcat(s1, s2, …, sN)` – concatenates strings. [9]

Digital Sound Effects Project

You are going to write a MATLAB program to read a wave audio file (*.wav) and modify the audio with an echo sound effect.

MATLAB Program

Write a MATLAB program to do the following. Your code must be commented.

1. Create `function echo_effect( filename, distance, alpha )`

   Inputs to the function:

```
% filename - wave file to read
% distance - distance in meters to reflection surface
% alpha    - reflection coefficient
```

2. Read a `*.wav` file using the `wavread` command.

```
[x, fs] = wavread(filename);
```

3. Calculate the time delay of the echo (in seconds). Assume the speed of sound at sea level in dry air at 20° C is 342.2 meters/sec. Use this time delay and the sampling rate (`fs`, returned by `wavread`), to calculate the echo delay in number of samples.

4. Modify the original audio signal with an echo effect, using a reflection coefficient of `alpha` and an echo delay in samples based on the distance to the reflection surface (as calculated in #3).

5. Graph the original sound and the modified sound. Use the `subplot` command to create a 2x1 figure and plot both graphs on the same figure.

6. Play the original sound and the modified sound using the `soundsc`command.

7. Write the modified signal to an output file using the `wavwrite` command.

```
[path, name, ext] = fileparts(filename);
wavwrite(y, fs, strcat(name, '_echo', ext));
```

8. Create a `*.wav` file using Audacity or another recording tool. Record and save a few seconds of your own speech.

9. Call your function with the following input arguments:
   - `filename = 'test.wav'` (or whatever you named your recording file)
   - `distance = 100`
   - `alpha    = 0.5`

Analysis

Include the figure output from your program. Consider the following in your analysis.

1. Compare the graphs of the original speech signal and the modified speech signal. What changes do you see in the graph of the modified speech?

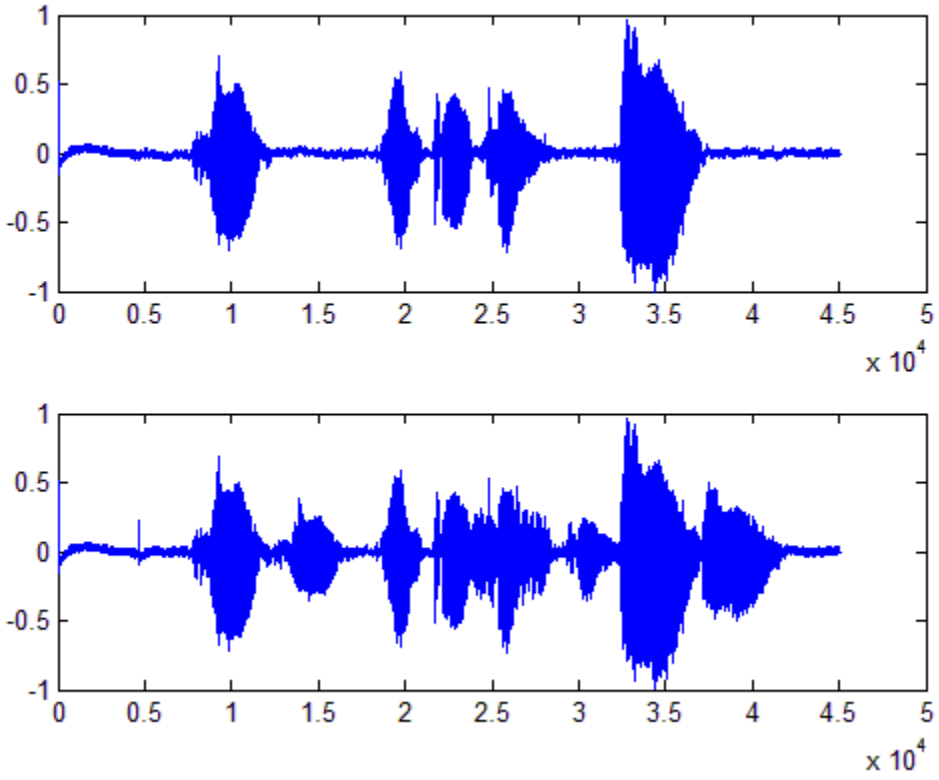
2. The following web page contains a table of absorption coefficients of common building materials. http://www.sae.edu/reference_material/pages/Coefficient Chart.htm. The reflection coefficient is the opposite of the absorption coefficient, energy that is not absorbed is reflected. The reflection coefficient indicates the ratio of acoustic energy that has been reflected. Use the table to compute the reflection coefficient of theater seats (wood, empty) and seats (fabric-upholstered, fully occupied) at 1 kHz. Call your function using the same `distance = 50` for each reflection coefficient (`alpha`) and compare the results. Think about your last trip to the movies. Why are the seats upholstered and the walls covered with drapes?

3. Using the same reflection coefficient `alpha = 0.5`, call your function with a `distance = 10` and `distance = 100` and compare the results. Interpret your results in terms of a real world situation.

Sound Effects Project – Sample Code

```
functionecho_effect( filename, distance, alpha)
% filename - wave file to read
% distance - distance in meters to reflection surface
% alpha    - reflection coefficient
%
% echo_effect('test.wav', 20, 0.5);

closeall;

% speed of sound at sea level (m/s) and 20 deg c
speed_sound = 343.2;

[x, fs] = wavread(filename);

% time in sec of ech
time_echo = (2 * distance) / speed_sound;

% echo delay in samples
samples_echo = fix(time_echo * fs);

% apply echo
for n = 1:length(x)
if (n <= samples_echo)
y(n) = x(n);
else
y(n) = alpha * x(n-samples_echo) + x(n);
end
end

figure;
subplot(2,1,1);
plot(x);

subplot(2,1,2);
plot(y);

%soundsc(x, fs);
soundsc(y, fs);

% write the echo to save file
[path, name, ext] = fileparts(filename);
wavwrite(y, fs, strcat(name, '_echo', ext));

end
```

Sound Effects Project - Sample Output and Analysis



1. The graph of the modified speech signal contains a replicated copy of the original speech, but with less amplitude. This is the echo of the speech – the same speech is repeated at a later time and with less amplitude. It is most obvious at the end of the graph because the echo is not overlapping original speech.

2. Theater seats (wood, empty) have an absorption coefficient of 0.05 at 1 kHz, so reflection coefficient alpha = 1 - 0.05 = 0.95. Seats (fabric-upholstered, fully occupied) have an absorption coefficient of 0.88, so reflection coefficient alpha = 1 – 0.88 = 0.12 [10]. The echo is barely noticeable for alpha = 0.12 (seats, fabric, full), whereas the echo is very noticeable for alpha = 0.95 (seats, wood, empty). Theaters use upholstered seats and drape covered walls (high absorption coefficient, low reflection coefficient) to reduce the echo of the sound in the room.

3. The greater this distance, the longer the delay between the original speech and the echoed speech. For example, an echo in large open building will have a much greater delay time than an echo in a small room.

Lesson Plan 4: SASE Lab (Speech Analysis and Sound Effects)

NAMES: J. Rebecca Dowell and Jacob Zurasky

CONTENT AREA: Pre-Calculus

GRADE LEVEL: 9-12

TOPIC: Modeling Sound with Functions

NEXT GENERATION SUNSHINE STATE STANDARDS:

MA.912.A.2.2 Interpret a graph representing a real-world situation.

COMMON CORE STATE STANDARD MATHEMATICS:

MACC.912.F-IF.4 For a function that models a relationship between two quantities, interpret key features of graphs and tables in terms of the quantities.

COMMON CORE MATHEMATICAL PRACTICE:
Use appropriate tools strategically.

UNIT: Speech Processing

GOAL: Student will understand how spectrogram output differs for different input sounds. Students will appreciate or realize speech processing as a real-world application of mathematics.

OBJECTIVE: Students will use SASE Lab tool analyze different speech inputs. Students will make conjectures on how to interpret spectrogram graphs.

MATERIALS: Computer with sound system (microphone and speakers). MATLAB software.SASE Lab code. Optional materials: musical instruments or Internet access to You Tube.

PRIOR KNOWLEDGE:Students should have basic knowledge of using MATLAB programs. Students should have basic knowledge of sound and speech processing, and should have completed the previous three Speech Processing Unit lessons.

PROCEDURES: This lesson is the culmination of the previous three lessons. Students experiment with the SASE Lab software as a guided exploration and inquiry. During the *SASE Lab Guided Inquiry*, students make connections to what they learned in the previous lessons. Refer to the *AEGIS RET Speech Processing Technical Report* [1] for more information. This paper gives an overview of speech processing and then goes into further detail on each stage of speech processing. The "Experiments and Results" section describes the SASE Lab software and should be read by the teacher prior to implementing the *SASE Lab Guided Inquiry.*

EVALUATION: *SASE Lab Guided Inquiry*

REFERENCES:

[1] Zurasky, Jacob. *AEGIS RET Speech Processing Technical Report.* Florida Institute of Technology, July 2012.

SASE Lab Guided Inquiry

SASE (Speech Analysis and Sound Effects) Lab is a tool used to analyze the stages of speech processing. Work through the steps outlined below with a partner. Record your results as you work.

1. Use the "Start Recording" and "Stop Recording" buttons to record yourself saying "*a e i o u*".

2. Save your speech as a wave file (File - Save), using your name in the filename.

3. Play your recorded speech (use "Play" button).

4. The "Speech Signal" graph (upper left) displays the graph of your speech, sample number vs. amplitude. How can you identify the speech "a e i o u" in the graph? What is happening in the graph in sections where you are not speaking?

5. On the "Speech Signal" graph, click on a section of speech to select a frame of speech to be processed. The remaining five plots now show the selected speech frame at the various stages of speech processing.

6. A spectrogram displays how the frequency content of the signal changes over time. From the menu, choose View – Spectrogram. The top graph displays the graph of the original speech signal, and the bottom graph displays the spectrogram of your speech. How can you identify the speech "a e i o u" in the graph? Compare/contrast the speech patterns for each vowel in the spectrogram. What is happening in the spectrogram in sections where you are not speaking?

7. If your speech is greater than 3 seconds long, use the scroll bar in the middle of the screen to select a 3 seconds sample of your speech. A new window opens containing the graphs of the selected speech. You may also use the "Play Section" button to play the selected 3 seconds of speech. You may use the scroll button to select a different 3 second sample.

8. Compare you spectrogram with your partners. How are they the same? How are they different? How do you think the computer uses the spectrogram frequency information to recognize speech? Recognize speakers?

9. Thinking back to the Frequency Analysis lesson, what mathematical process do you think the SASE Lab program uses to produce the frequency information from the speech input signal?

10. Experiment with different sounds and record and interpret your results in terms of the spectrogram output.

    a. Different sounds / different simple phrases
    b. High pitch vs. low pitch
    c. Whispering vs. talking loudly
    d. Silence
    e. Whistling

11. If you have an instrument, record yourself playing a few notes. Otherwise, go to YouTube and search for "flute music" or "violin music" and record a sample. Video "lessons" usually have simpler music. View the spectrogram, record and explain your results. Thinking back to the Sound of a Sine Wave lesson, how does the spectrogram reflect the different musical notes played?

12. Reopen (File – Open) your previous wave file of the vowel sounds and experiment with the sound effects.

13. Apply each of the sound effects. For each effect, play the modified sound and view the spectrogram. How does the spectrogram change for each sound effect, and what do these changes signify? Each sound effect has parameters you can modify with the scroll bars in the upper right.

    a. Echo
    b. Reverb
    c. Flanger
    d. Chours
    e. Vibrato
    f. Tremolo
    g. Voice Changer

14. In the previous lesson, Digital Sound Effects, you wrote a program to apply an echo sound effect. Now that you have written echo, how do you think you would modify your program to apply a reverb effect? Compare the two spectrograms for ideas. Do you have any idea how to produce any of other sound effects?

Speech Processing Report

Conclusion

In paragraph form, summarize the four lessons we have completed.

Following your summary, reflect on your learning and your experiences during the Speech Processing lessons. Consider the following questions:

- What did you learn from this project?
- Do you feel this project furthered your understanding of applications of trigonometric functions? How does this compare to studying applications of trig functions with word problems from the textbook?
- State any "lessons learned". What specifically would you do differently if you were to do this project over again? How would you improve or extend your project?
- From the four lessons completed, which was your "favorite" lesson and why?
- What grade would you give yourself on this project?
- What did you think about learning and using MATLAB? Would you like to use MATLAB again? Do you have any ideas for how you would like to use MATLAB again in the future?
- Do you have any "research questions" you developed throughout the project? What would you like to learn more about?
- Do you have any "new ideas" for applications of speech processing?
- Would you be interested in doing more "real world applications" projects like this in the future?
- Has this project influenced your interest or changed your mind about studying science, technology, engineering, and mathematics in the future?
- Has your viewpoint on the role of mathematics in the "real world" changed as a result of this project?Students often ask me "When am I evergonna use this in real-life?" Do you think this question has been addressed through this project?
- Do you have any suggestions to me for improving these lessons for use in future classes?

Report

You may choose to format your project as a report and/or poster that is suitable for classroom display. Points may be added or deducted for neatness/presentation. Your project must include the following items:

- Title page (include a project title, student names, class, date).
- MATLAB Projects – hard copy of code and analysis for each lesson.
- Conclusion summary and reflection.