ECE 321 Spring 2014
Matlab Project 1
Due February 27, 2014 6PM

The goal of this project is for you to work in groups of 3 (max of 4) to synthesize a specified melody. Over the course of the semester, you will revisit this melody, modifying it and eventually applying all you've learned to a melody of your own choice.

Each group will email their submission as a .zip file entitled *P1Name1Name2Name3.zip* where *Name1*, *Name2* and *Name3* are the last names of the group members. Make sure the .zip file includes all required files. For Project 1, these include:
the .mat file ***music.mat*** in which you've saved all vectors generated as part of this project,
the mfile ***Project1.m***,
the functions ***gentone.m***, ***genrest.m*** and ***shape.m***.
A .doc file ***p1Name1Name2Name3.doc*** which is your project report, containing the requested plots and your answers to the questions posed below. Make sure everything is clearly annotated both for which part of the assignment it's answering and that it's clear to anyone what the graphs are depicting and what their axes are.

You will need to be able to listen to the sounds you generate to make sure you've done the project correctly. If you work in 214, that will require headphones. Also, you may not be able to listen to sounds created using remote desktop unless you export them to your local machine.

The project is to be submitted electronically to me (kpayton@umassd.edu) by Thursday February 27, 6PM. Be sure each plot is clearly labeled both in terms of what it represents and axis labels. Make sure one group member takes responsibility for writing a section and another member checks the work. **Each .m file should have an author and a checker included in the file's comments.** Also make sure that the work is distributed fairly (e.g. one person shouldn't write all the code). The same goes for the report. Each of you should take responsibility for specific sections of the report and someone else should check sections and document that in your report.

Background
First, you will explore how to synthesize tones that correspond to musical notes. Then you will construct the first few bars of Beethoven's 5^{th} symphony in C-minor. No background in music is required. If something is unclear, please ask.

Musical notes can be synthesized using a sinusoid whose frequency determines the note pitch.

Musical notes are arranged in groups of twelve, called octaves. An octave change in frequency corresponds to a doubling of the frequency. The notes we'll use in Project 1 are in the octave spanning 220 Hz to 440 Hz. The twelve notes in an octave are logarithmically spaced in frequency with each note being $2^{1/12}$ times the frequency of the next lowest note.

The table below shows the ordering of notes, some of which will be used to synthesize the music for the first part of the project as well as the fundamental frequencies for some of those notes:

Table 1

Note	Frequency (Hz)
A (low)	220
$A^{\#}$, B^{b}	$220*2^{1/12}$ =
B	$220*2^{2/12}$ =
C	
$C^{\#}$, D^{b}	
D	
$D^{\#}$, E^{b}	
E	
F	
$F^{\#}$, G^{b}	
G	
$G^{\#}$, A^{b}	$220*2^{11/12}$ =
A (high)	440

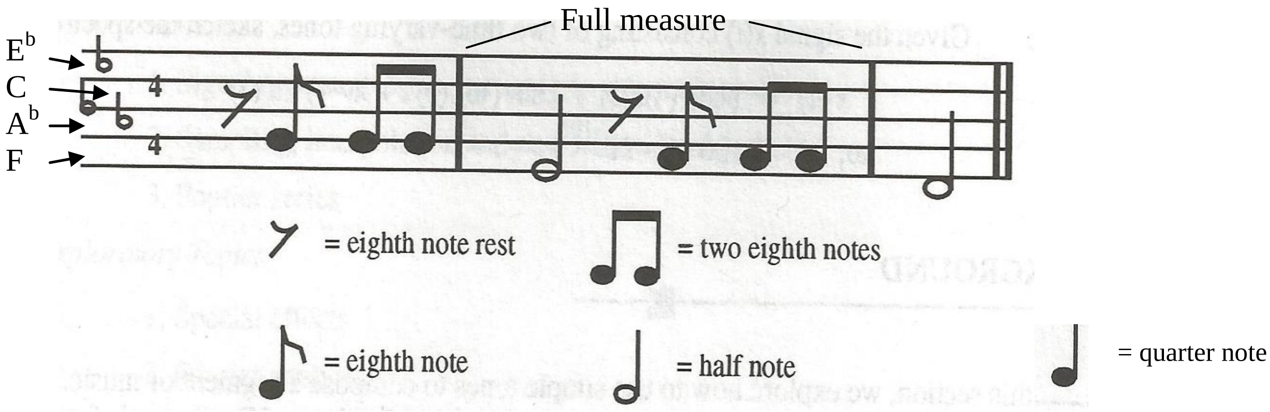
The sharps (#) and flats (*b*) listed on the same line are the same frequency. You will need to complete this table in order to know which frequencies to synthesize. **The completed table should be included in your .doc report file.**

A musical score instructs the musician what notes to play and how to play them. Vertical placement indicates which note; from low to high, the horizontal lines on the score correspond to E, G B, D then F. The spaces correspond to F, A (high A from Table 1), C then E. Sharps and flats that apply to the entire row are indicated in the far left.

Our score is shown on the next page. Note that, since the high E has a flat mark (b) every E is played as E^b, in particular, the low E is also an E^b.

The sequence indicates the order in which the notes are to be played. The kind of note indicates how long it should be played (e.g. a quarter note is twice the duration of an eighth note). The 4 over 4 on the left indicates there are four quarter note equivalents per measure, summing to a whole note, and each quarter note equivalent gets one beat. A measure refers to the interval between vertical lines. Rests indicate when (and how long) **not** to play any notes.

For our score, a beat (quarter note) lasts approximately ½ second. The first, partial, measure has an eighth note rest then 3 eighth notes. The second, full, measure has a half note, an eighth note rest then 3 eighth notes (1/2 + 1/8 + 1/8 +1/8 +1/8 = 1).



In the following table, are the durations in seconds of some notes/rests. **In your report, you should include a completed version of this table:**

Table 2

Note/Rest	Duration in sec
Whole	
Half	
Quarter	1/2
Eighth	
Sixteenth	

Note Generation:
In the simplest case, each note may be represented by a sinusoidal tone burst followed by a very short period of silence (a pause, not a musical rest). The pause allows us to distinguish between separate notes at the same pitch and should be the same for all notes, independent of the note duration. A pause of about 0.1 sec is probably reasonable.

Part 1: Basic melody generation
First, you will generate each of the required notes and rests. There are a total of 4 unique notes and one rest in this melody. Create the script file **Project1.m** in Matlab. Make sure it includes all the steps you use to generate the notes, melody and plots, who wrote it and who checked it.

The sampling frequency you will use is 8192 Hz (i.e. your time sequence will be T=[0:1/8192:S] where S is the number of seconds of sound you want to synthesize).

Task 1: Write a function, **gentone.m**, that generates a note of specified duration value (in seconds), frequency (in Hertz) and gain (start with `gain` = 1) and includes the necessary pause at the end. The function should have the following syntax:

```
note = gentone(frequency,duration,gain)
```

Task 2: Write a function, **genrest.m**, that generates a rest of specified duration value (in seconds). It should have the following syntax: `rest = genrest(duration)`

Task 3: In your mfile script, **Project1.m**, use your functions to generate each note and rest in the score; saving each to a variable name that indicates what the note or rest is (e.g. half_C or quarter_rest). **Plot each of the 4 notes vs time in sec in separate plots (you can use subplots). Use these plots to confirm your notes have the correct durations. Also, plot two periods of each note and confirm they are the correct frequencies. Include all plots, well labeled, in your report. You also need to include appropriate text to indicate that you checked (and how you checked) both durations and frequencies.**

Task 4: In your **Project1.m** mfile script, create your melody by concatenating the notes and rests in the correct order. Concatenation is achieved as in the following example where the signals are all assumed to be row vectors:

```
melody = [note1 note2 rest 1 note3];
```

If your signals are column vectors then you need a semicolon between signals.

You can hear your melody by using the soundsc command:

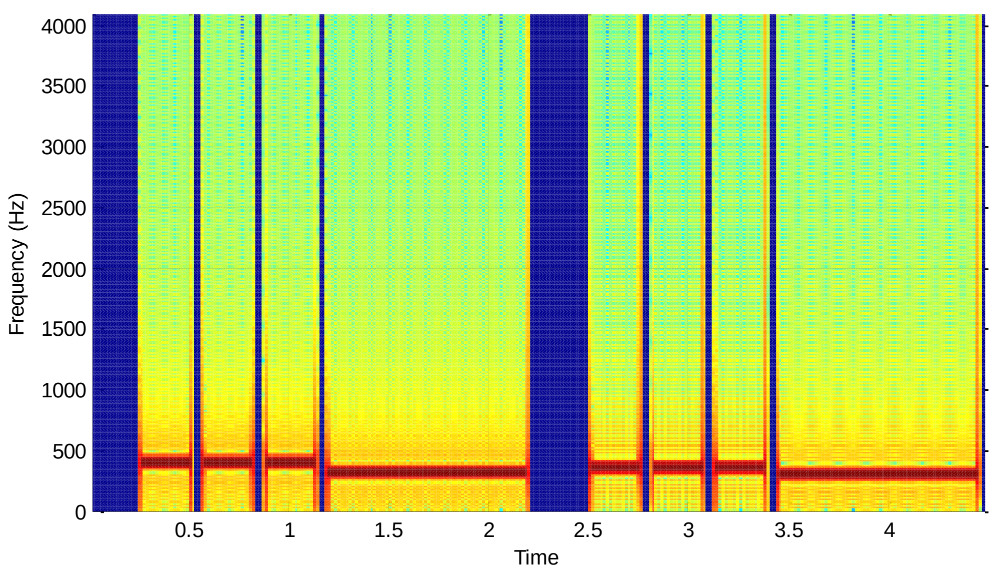
```
soundsc(melody);
```

Compare your melody with one I generated. First, listen to the .wav file: **melody.wav** included in this directory (just use your machine's default audio app). Does it sound the same as your melody? **Comment on any differences you hear in your report.** Second, use the following command to plot something called a spectrogram:

```
spectrogram(melody,256,196, 512 , 8192,'yaxis')
```

Reminder: when you submit your report, include all plots, clearly labeled, in your .doc file.

The spectrogram of my melody is shown below:

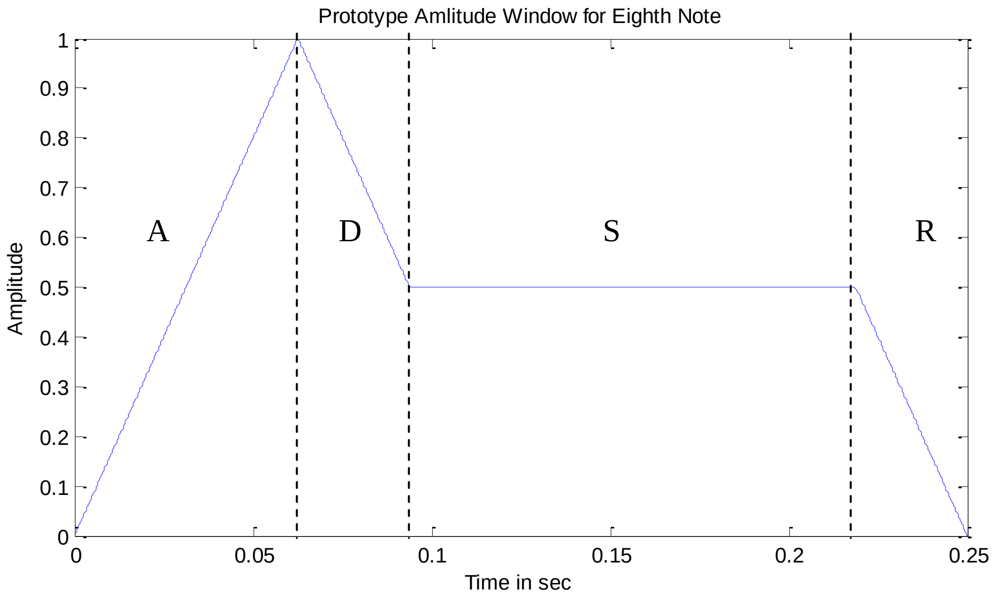


This spectrogram plots frequency on the vertical axis, time on the horizontal axis and magnitude as color (red is large magnitude, blue is small). Notice its similarity to the musical score. The two wide blue bands correspond to the two rests. The horizontal red bars correspond to the notes. Low notes are displayed toward the bottom of the plot and high notes are toward the top of the plot. All notes synthesized are below 500 Hz. Notice that the plot goes up to 4000 Hz. That is the highest frequency you should try to synthesize at this sampling rate (8192 Hz).

Part 2: Improving the quality of your melody
There are several things you can do to improve the perceived naturalness of your melody. For this first assignment we will focus on amplitude and reverberation.

Task 1: Modify the amplitude profile. You might have noticed clicks at the beginning and end of each of your notes. These clicks are due to the sudden onset of the tones. Naturally-produced notes have non-uniform amplitudes over the duration of the note. You are going to apply that concept to your melody.

Write a function, **shape.m**, that will generate an amplitude contour (also known as a window) that looks something like the following:



The variation over time is divided into four segments: Attack (A), Decay (D), Sustain (S) and Release (R).

Start by keeping the Attack, Decay and Release durations constant for all notes (A ~ .06s, D ~ .03s and R ~ .03s); only vary the Sustain duration for different length notes. You get to decide exactly what the durations should be for your segments. **List your chosen values in your report.** In addition to the required shape, above, you also may optionally allow your function to accept an additional parameter that modifies the shape of your A, D, S and R segments. Besides straight lines as shown, you may consider replacing D, S and R with a single decaying exponential (typical of plucked, strummed or hammered instruments).

Modify your gentone function to call your shape function. You should add a parameter to the gentone syntax as follows:

```
note = gentone(frequency,duration,gain,contour)
```

such that: a) if the contour parameter is zero, no shape is applied, b) if the contour parameter is 1, the shape shown in the figure above is applied. If you wish to create an additional contour, use 2 to specify it.

To apply the contour to a tone, make sure your code generates exactly the same number of samples as your tone then do an element-by-element multiply (.*) within your gentone function. Also, both the tone and the contour must be either row vectors or column vectors.

In your report, plot your contour vs time for an eighth note and a shaped eighth note (either frequency, just specify which).

Generate your melody with all notes "shaped". P**lot the shaped melody vs time and the spectrogram of your shaped melody.**

A final reminder about amplitudes: You can change the relative amplitudes of different notes with the gain parameter.

Part 3: Reverberation
The concert hall in which music is played contributes to the overall perception of the music. For this project, you will simulate a very simple echo environment. In your Project1.m script, create a vector of zeros called **echo** that is 2 seconds long.

Next, put the following values at the designated times in the vector:

Time (s)	Sample Amplitude
0	1
.5	.5
1	.25
1.5	.125
2	.0625

Now, convolve your shaped melody from Part 2 with the echo vector and call the result **reverb**. Reminder: In order for conv.m to work correctly; both vectors must be either row vectors or column vectors. Listen to the reverberant melody. **Plot reverb vs time and plot its spectrogram.**

Include the following in your report:

1. Write a mathematical expression for the melody in Part 1. Your answer should be written as an algebraic combination of unit steps and sinusoids and take into account both rests and pauses.

2. Describe the ways in which the spectrogram of the reverberant melody in Part 3 differs from the original melody in Part 1. Specifically discuss effects of the amplitude and reverberation modifications you made on the spectrogram.

3. Describe how the modifications in Parts 2 and 3 changed the way the melody sounded. Relate the differences you hear to the amplitude and reverberation modifications you made.