Matlab 1

Sunday, February 21, 2021 10:30 PM

321Project…
(2)

321Project…
(2).docx

ECE 321 Spring 2021
Matlab Project 1 (40 points possible)

The goal of this project is for you to synthesize a specific melody. Over the course of the semester, you will revisit this melody in Projects 2 and 3, modifying it and eventually applying all you've learned to a melody of your own choice.

You will need to be able to listen to the sounds you generate to make sure you've done the project correctly. Use headphones when working in common areas so you don't disturb those around you.

Each group (max size of 3) will submit a .zip file entitled *P1GroupX.zip* where *X* is the group number in myCourses. Make sure the .zip file includes ALL required files. For Project 1, these include:

- the script ***Project1.m*** which includes the code used to call all functions and create all plots,
- a published .pdf version of that script that must include the plots,
- the functions ***gentone.m***, ***genrest.m***, ***shape.m***
- a .docx or .pdf file ***p1GroupX***, that is your project report. It must contain the requested tables, all plots with explanations why those plots show what was requested and your answers to the questions posed below. You may use the ECE Lab Report Guidelines to structure your report. On the cover sheet include the names of all group members who contributed.

Make sure graphs, functions and answers are clearly annotated to indicate which part of the assignment they are answering. Graphs must be labeled/commented so it's clear to anyone what they are depicting and what their axes are.

The project is to be submitted electronically to MyCourses by the deadline. Zip ALL relevant files into a single .zip file. Click on the assignment title then browse your local computer to upload the correct .zip file. Make sure one group member takes responsibility for writing a section and another member checks the work. 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. If your code doesn't run when I download it, I will give you a zero and one opportunity to resubmit with a 5-point penalty. When this happens, its often because a necessary file was left out of the .zip.

To assist you in debugging your code, I have created "problems" on the Mathworks Matlab Grader site. Click the link in the email you receive and log in with/create your MathWorks Account using your UMass Dartmouth email. The problems are designed to check some of your variables and functions and let you know if they are correct. You do not have to use them but students have found it helpful to confirm their code is working correctly. I will check the site to see what kinds of issues students are having.

In addition to the group submission, each group member will need to complete and submit a team member evaluation form. Your grade on the project will be adjusted based on the team evaluations.

Part 1: 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 5th 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 increase in frequency corresponds to a doubling of the frequency. All the notes we'll use in Project 1 are in the octave spanning 220 Hz to 440 Hz. The twelve (western scale) 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 the 13 notes in our frequency range of interest. For Project 1, you will need to synthesize 4 of these notes.

Table 1

Note	Frequency (Hz)
A (low)	220
$A^{\#}$, B^{b}	$220*2^{1/12} = ?$ 233.08
B	$220*2^{2/12} = ?$ 246.94
C	? 261.63
$C^{\#}$, D^{b}	? 277.18
D	? 293.66
$D^{\#}$, E^{b}	? 311.13
E	? 329.63
F	? 349.23
$F^{\#}$, G^{b}	? 369.99
G	? 392
$G^{\#}$, A^{b}	$220*2^{11/12} = ?$ 415.3
A (high)	440

The sharps (#) and flats (*b*) listed on the same line correspond to the same frequency (i.e. A# is the same as B^{b}).

Task 1: Complete this table in order and include it in your .docx report file (1 point).

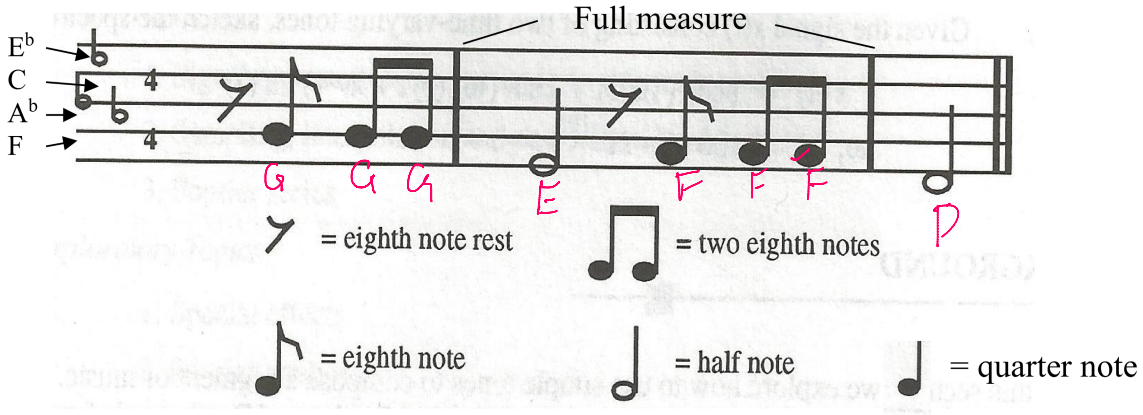
A musical score instructs the musician what notes to play and how to play them. Vertical placement indicates which note; from low to high, the horizontal lines on the score correspond to E, G B, D then F. The spaces correspond to F, A (high A from Table 1), C then E. Sharps and flats that apply to the 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}** even though it is not explicitly labeled as such.

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.
Task 2: Include a completed version of this table in your report (1 point):

Table 2

Note/Rest	Duration in sec
Whole	2.0
Half	? 1
Quarter	0.5
Eighth	? 0.25
Sixteenth	? 0.125

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 frequency and should be the same for all notes, independent of the note duration. For now, specify a pause of 0.03 sec.

Part 2: Basic melody generation (16 points)
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, **genrest.m**, that generates a rest of specified duration value (in seconds). It should have the following syntax:

```
rest = genrest(duration)
```

Task 2: Write a function, **gentone.m**, that generates a note of specified duration value (in seconds), frequency (in Hertz) and gain and includes the necessary pause at the end. The sinusoid plus pause should total the required note length. You may call **genrest** from within **gentone** to generate the pause then append the pause to the end of the note OR set the final 0.03 s worth of samples to zero directly. The **gentone** function should have the following syntax:

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

Task 3: Start an mfile script, **Project1.m**, that uses your functions to generate each note and rest in the score. Save each note to a variable name that indicates what the note or rest is (e.g. half_C or quarter_rest).

Task 4: The script should then **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.** The script should then, **plot exactly two periods of each note vs time in separate subplots. Confirm each note has the correct period. 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 periods.**

Task 5: Next 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 rest1 … noteN];
```

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);
```

Task 6: Compare your melody with one I generated two ways. **First**, listen to the .wav file: **melody.wav** included in the Matlab Projects content area (you can use your machine's default audio app or Audacity). 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')
```

The spectrogram of my melody is shown below:

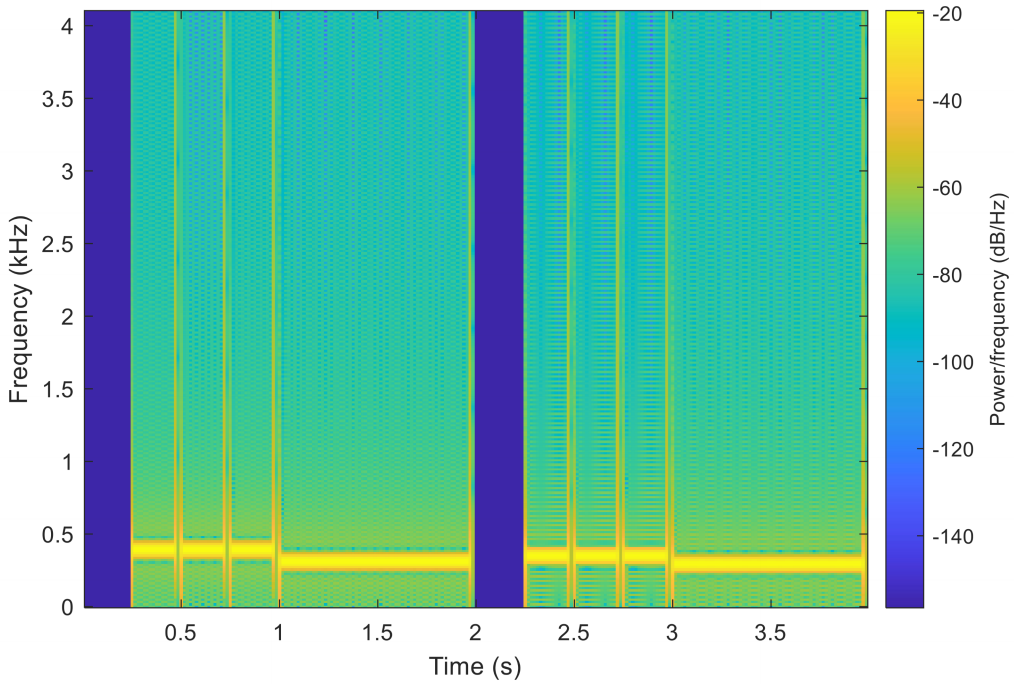


Figure 1: *Spectrogram of Project 1 melody. Frequency is plotted on the vertical axis, time on the horizontal axis and magnitude as color (yellow is large magnitude, blue is silence). The two wide vertical blue bands correspond to the two rests. The very narrow blue bands correspond to the pauses between notes. The horizontal yellow bars correspond to the notes. Low notes are displayed closest to the bottom of the plot and high notes are toward the top of the plot. All notes synthesized for this project are below 500 Hz. Notice the similarity of the region below 500 Hz to the musical score.*

This spectrogram displays frequencies up to 4000 Hz. That is the highest frequency you should try to synthesize at this sampling rate (8192 Hz). **Describe how your spectrogram compares to the one above (similarities and differences).**

Part 3: Improving the quality of your melody (7 points)
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.

You might have noticed clicks at the beginning and end of each of your notes. These clicks are due to the sudden start 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.

Task 1: Create an amplitude profile: Write a function, **shape.m**, that generates an amplitude contour (also known as a window) that looks something like the following:

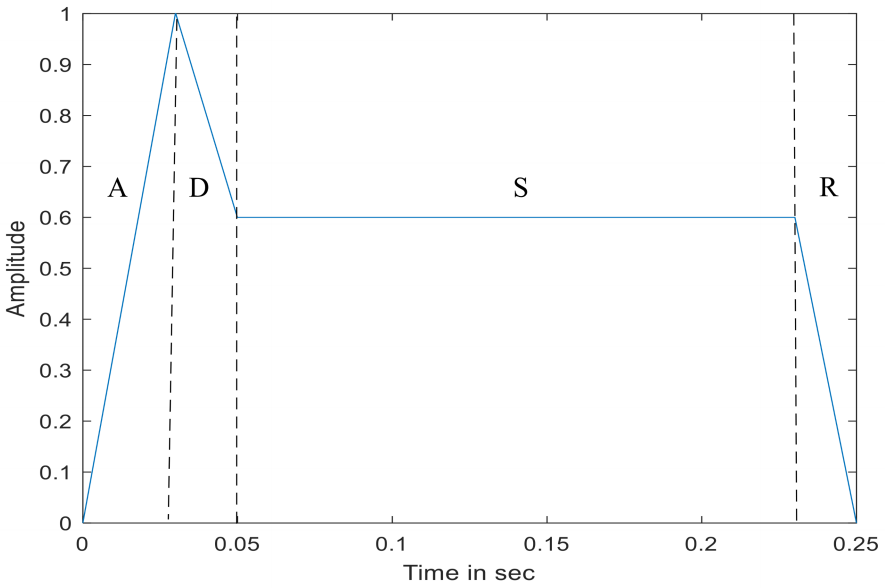


Figure 2: *Example plot of the envelope generated by the function shape. The envelope is made up of four segments: Attack (A), Decay (D), Sustain (S) and Release (R). Attack is 0.03 s long, rising from 0 to 1. Decay is .02 s long and falls from 1 to 0.6. Sustain remains at 0.6 until the note length minus Release. Release falls from 0.6 to 0 over the last 0.02 s.*

Keep the Attack, Decay and Release durations constant for all notes. Start with A = .03s, D = .02s and R = .02s. Sustain duration is the only thing that should change for different length notes.

Task 2: Generate a plot of your shape function for a quarter note. Once you have generated the shape plot for your report, you may modify the duration of each segment if you wish.

Task 3: Modify your gentone function to call your **shape** function (rename it to **gentone1** so you don't get confused). You should add a parameter to the **gentone** syntax as follows:

```
note = gentone1(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. You may eliminate the pauses used in Part 2 when contour is not zero.

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 **gentone1** function. Also, both the tone and the contour must be either row vectors or column vectors.

Task 4: Plot a shaped eighth G note using gentone1 from Task 3.

Task 5: Generate your melody with all notes "shaped".

Task 6: P**lot the shaped melody vs time and the spectrogram of your shaped melody.**

A comment about the gain term: You can change the relative amplitudes of different notes with the gain parameter if you wish.

Part 4: Reverberation (5 points)
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.

Task 1: 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

Task 2: Convolve your shaped melody from Part 3 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.

Task 3: Plot reverb vs time and plot its spectrogram.

Include the following in your report:

1. **(6 points)** Write a mathematical expression for the melody synthesized in **Part 2**. Your answer should be written as an algebraic combination of analog unit steps and sinusoids. It must take into account both the rests and brief pauses. You must include the onset time of each sinusoid, its duration and frequency in the mathematical expression.

2. **(2 points)** Describe the ways in which the spectrogram of the reverberant shaped melody in Part 4 differs from the original melody in Part 2. Specifically discuss effects of the amplitude and reverberation modifications you made on the spectrogram.

3. **(2 points)** Describe how the modifications in Parts 3 and 4 changed the way the melody sounded. Relate the differences you hear to the amplitude and reverberation modifications you made.