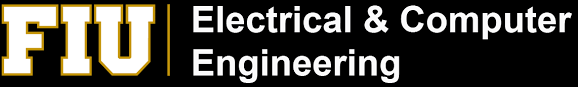
****

**ELECTRICAL AND COMPUTER ENGINEERING**

FIU College of Engineering and Computing

**EEL 3135: SIGNALS AND SYSTEMS**

|  |  |  |
| --- | --- | --- |
| Project Title | **Lab 04: Synthesis of Sinusoidal Signals—Music Synthesis** | |
| **Name:** | **Richard** | **Hernandez** |
| Date | **12/4/19** | |

**Honor Pledge:**

"*On my honor, I have neither given nor received aid on this assignment.*"



|  |  |
| --- | --- |
| ***For Official Use Only*** | |
| ***Comments:*** | ***Grade / Score:*** |
| ***Graded by:*** |

1. OBJECTIVE

Music is a common example of signals, and a good way to practice the knowledge acquired about signals and systems in class. The main objective of this project is to learn and at the same time apply this formulas and systems into a real-life situation. By using signals with different frequencies and adding them up to acquire a new signal that will make an appealing sound when amplified by a system in this case the soundcard. The final result will be a MATLAB function that will synthesis a song by using the frequencies, amplitude, duration, other properties of the song.

1. WARM-UP and LAB EXERCISE

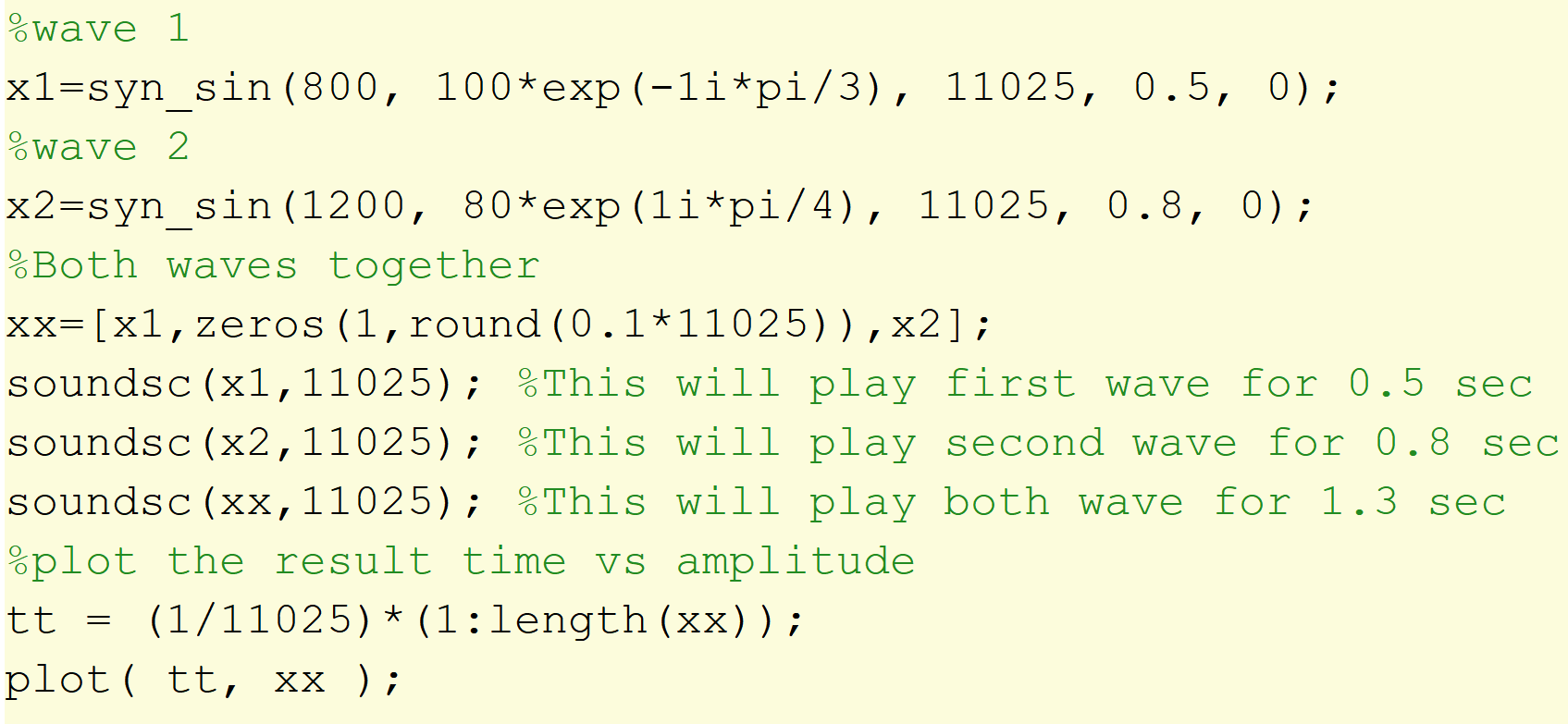
Section 2.2 **D-to-A Conversion:**

Because we are using a digital format we need to convert the signal of the music to analog, this will be done by the MATLAB function soundsc(). This function needs a vector and the advantage of using it is that the vector will be scaled automatically.

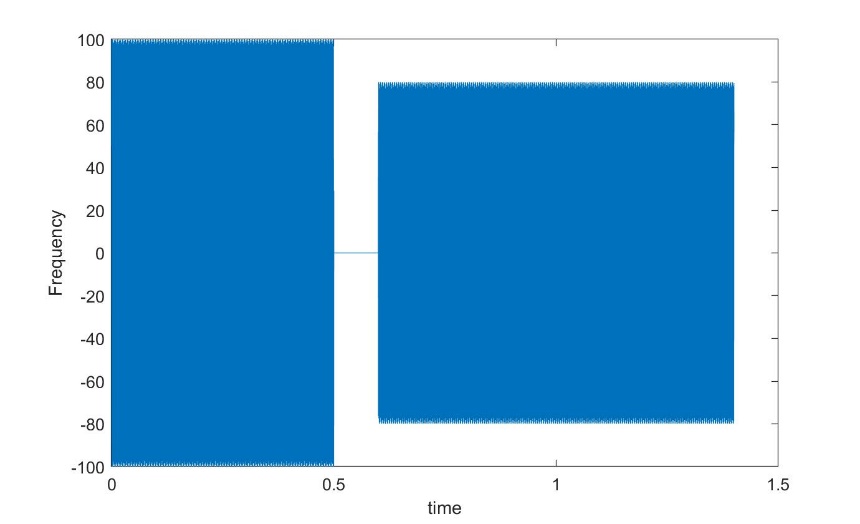
Now, during this section will use the function syn\_sin() which is given by Dr. Bai which will add synthesize cosine waves. This will create a wave, by using the parameters.

*Appendix: For the code of this function*

Next, task will be to create 2 waves using syn\_sin() called x1 and x2 and then unite them by concatenating them into a bigger vector called xx. The numbers used in the parameter were acquired from that section .



**Plot Result:**



Section 2.3 **Structures in MATLAB:**

A really smart way to represent the waves is by using MATLAB structures which can help us group the information of a wave in one variable. By using a dot after the variable we can add a property for example when a wave named “x”:

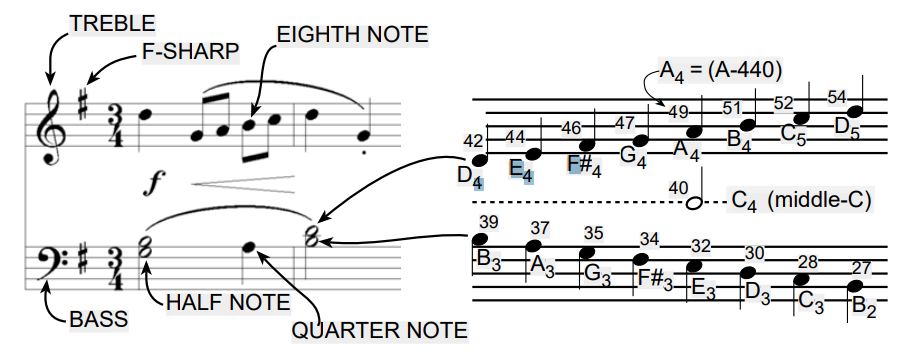
x.Amp = 5; --- Set amplitude of wave x to 5

x.phase = -pi/3 ---- set phase

This is just two examples how we can add more information inside the same variable. In addition, the types stored in each structure can be changed[vectors, strings, numbers, etc]

Section 2.5 **Piano Keyboard**:

This section was about how Pianos keys are divided and how they can be helpful to implement a better way to get the notes by using an pattern. By using this patterns we can algorithmically get the frequency of the note by just giving as an input the note number. Meaning that when the computer is given a number by using a function we can get the frequency, amplitude, duration and other valuable information.



We can also use the ratio 2^(1/12) to calculate any frequency of the notes anywhere on the keyboard. On the other hand, musical notation is a time-frequency diagram. And the shape of the notes defines the duration of the note.

Warm up

Section 3.1 **Note Frequency Function**:

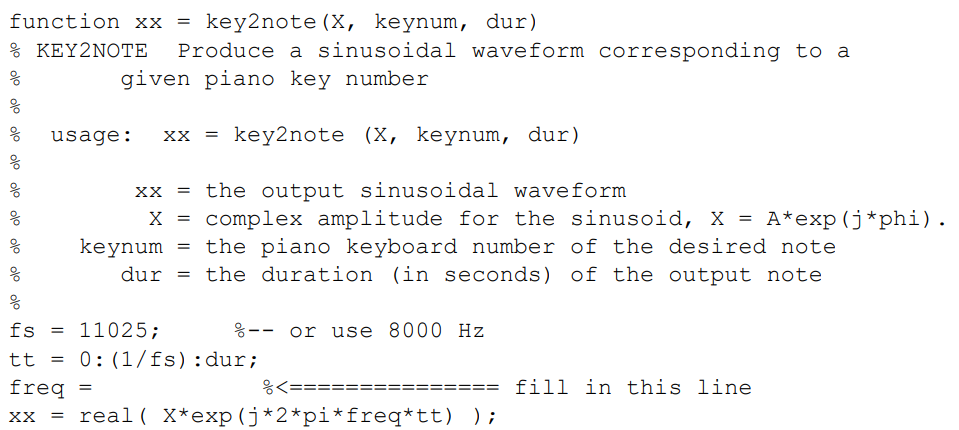
Creating a function to help in the creating of a waveform that will represent and sound as an specific key on the piano. The duration of this waveform will be given to the function as a parameter.

Function name: key2note

parameters: complex amplitude, number of the key, and duration of the note.

Returns a vector with the *real* part.

This is the code needed for this function but is missing the most important part since is missing the formula to calculate the frequency of the note.



By referencing the section before we can use the ratio and the key number to calculate the “*freq*” variable. This ratio will be multiplied by the 440 since that frequency of the A-440.

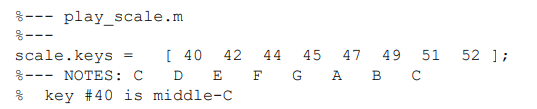
**freq = 440\*(2^((keynum-49)/12));**

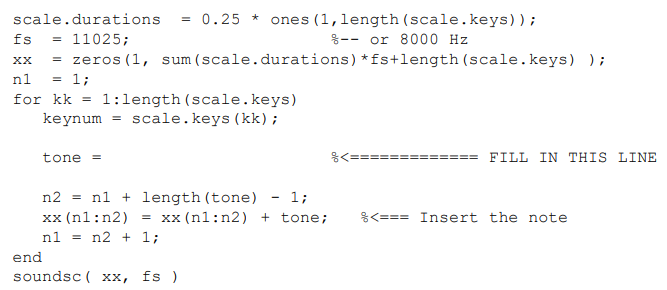
Now when this formula is added we now have a function that returns a note vector every time called using the required key and parameters. This gets us one step closer to the synthesis of the song.

Section 3.2 **Synthesize a Scale:**

Scale will play a series of keys one after the other, meaning that the use of key2note() is required to get the frequency of each of the notes. The next function will be called play\_scales() and there is no return value, and no parameters either.

The following code is given showing us the keys that will be pressed:





Steps of Code:

1. The code above will apply a duration to each note of 0.25 sec(QUARTER).
2. The sampling rate will be 11025 since it create a smooth sound and is typically use. After that “*xx*” will be use to make vector of 0 to represent the time of the whole scale.
3. After this a loop will go through every key from the scales.key adding the note information to the “xx” which uses n1:n2 as index to plot tones in correct places.
4. Lastly play the sound saved in “xx”.

Using the function to get the note.(*Maybe be change later to implement more features*)

**tone= key2note(X, keynum, scale.durations(kk));**

Section 3.3 **Spectrogram: Two M-files:**

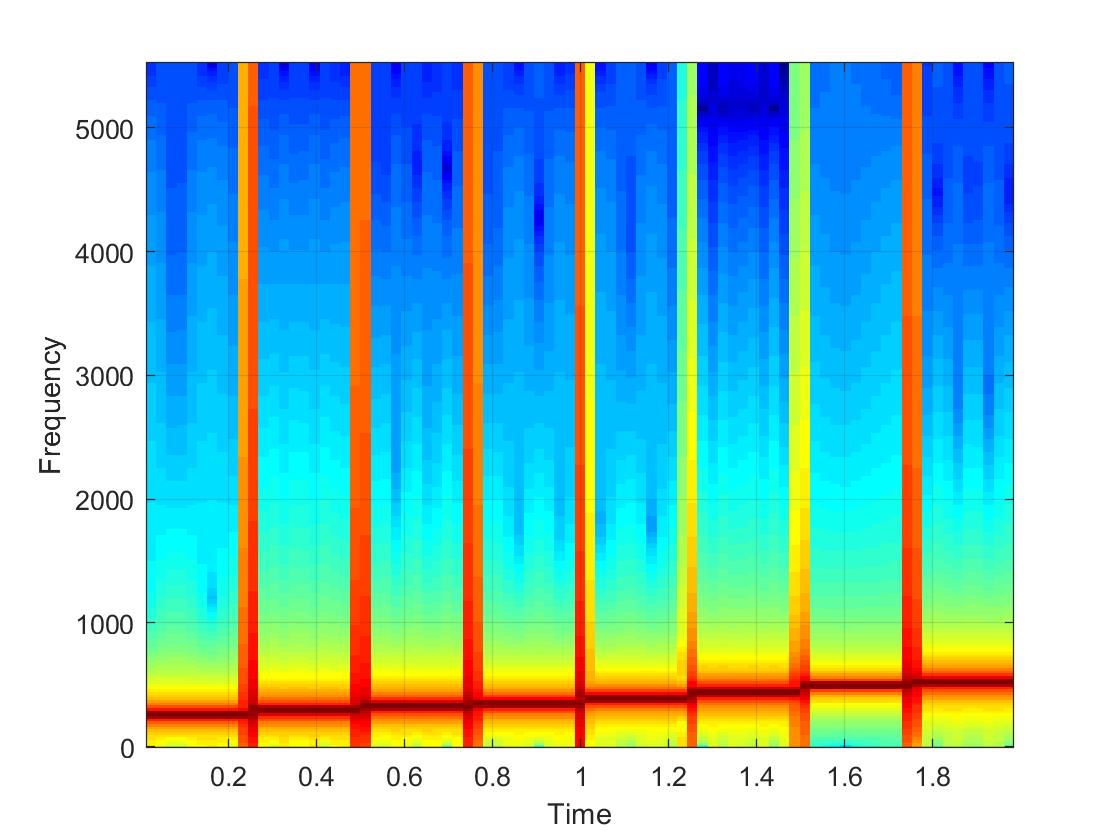
Part A:

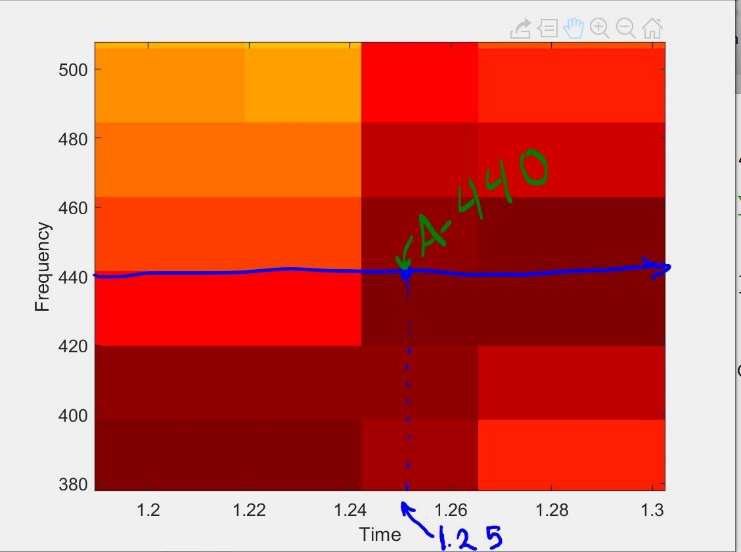
After the code added in section 3.2 the sound play one after the other.

Part B:

Now using the function specgram(xx,512,fs). The first parameter is the notes vector, second is the length of the window, and third is the sampling rate.

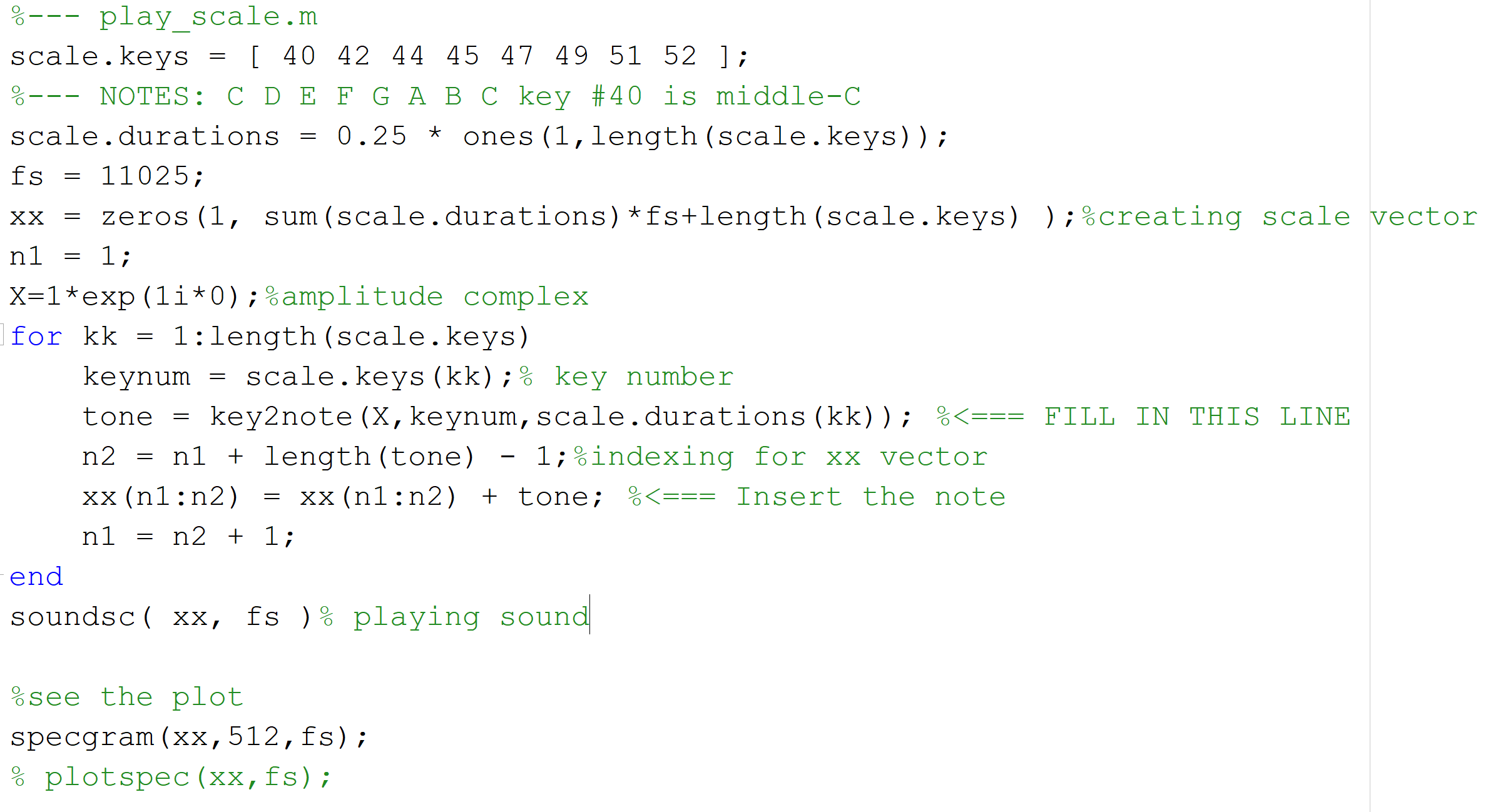
**Using specgram(xx,512,fs)**

Based on this image we can get information about what is happening.

* The time axis is 2 seconds long because we playing 8 notes for 0.25 seconds each.
* Also we can see the frequency increasing gradually each 0.25 sec.
* Between each note there is some kind of cut-off (this can be heard when the notes change).
* Here we can see a closer look at the frequency 440 Hz which is key number 49.
* The time of switch was 1.25 sec.

*Plotspec(xx,fs) is not currently installed in my version of MATLAB.*

Code:



*The code in this picture is the one used to play the scale and generate the specgram graph shown in this section.*

**Section 4 Lab Exercise: Synthesis of Musical Notes**

Section X.X: Part (Y)s

* *Answer the questions and cite the figures or the Matlab code if necessary.*
* *For eg: as seen in Fig.2 (a) the amplitude of the waveform is 1.6.*
* *The results of your experiments (plots, numeric values) should be under this heading.*
* *Write proper captions, x & y axis labels for each figure.*
* *Show the calculations neatly for each part.*

1. DISCUSSION

* *Describe your results and compare them to expected results or the theoretical values obtained through the calculations.*
* *Discuss some issues that you faced and how you overcame problems and made your code better*

1. APPENDIX

Section 2.2: **syn\_sin() by Dr Bai:**

function [xx,tt] = syn\_sin(fk, Xk, fs, dur, tstart)

%SYN\_SIN Function to synthesize a sum of cosine waves

% usage:

% [xx,tt] = syn\_sin(fk, Xk, fs, dur, tstart)

% fk = vector of frequencies

% (these could be negative or positive)

% Xk = vector of complex amplitudes: Amp\*eˆ(j\*phase)

% fs = the number of samples per second for the time axis

% dur = total time duration of the signal

% tstart = starting time (default is zero, if you make this input optional)

% xx = vector of sinusoidal values

% tt = vector of times, for the time axis

%

% Note: fk and Xk must be the same length.

% Xk(1) corresponds to frequency fk(1),

% Xk(2) corresponds to frequency fk(2), etc.

if nargin<5, tstart=0; end %--default value is zero

if length(fk)~=length(Xk), error('The vector length of freqeuncies and complex amplitudes should be the same!'); end

if ~isscalar(fs), error('The signal sampling rate ''fs'' should be a scalar!'); end

if ~isscalar(dur), error('The time duration of the signal ''dur'' should be a scalar!'); end

if ~isscalar(tstart), error('The starting time of the signal ''tstart'' should be a scalar!'); end

tt=tstart+0:1/fs:dur;

xx=zeros(1,length(tt));

for i=1:length(fk)

xx=xx+real(Xk(i)\*exp(1i\*2\*pi\*fk(i)\*tt));

end

* *Matlab code for Section X.X part (Y), the code should describe all the variables used and a brief description of what the code does*

Example:

*function xx = dtmfdial(keyNames,fs)*

*%DTMFDIAL Create a signal vector of tones which will dial a DTMF (Touch Tone) % telephone system.*

*%*

*% usage: xx = dtmfdial(keyNames,fs)*

*% keyNames = vector of characters containing valid key names*

*% fs = sampling frequency*

*% xx = signal vector that is the concatenation of DTMF tones.*

1. SELF VERIFICATION SHEET

* *If the section (or a part) is mainly done by Student A, Student B will verify the result and sign the sheet; and vice versa.*