EEE321\_Signals and Systems Lab-1

Sait Sarper Özaslan

22002861

Part 1

1. First one is a 4x1 matrix while the latter one is 4x1 matrix. As there is no semicolon, upon running the code it will show the output on command window.
2. This time, semicolon marks the end of statement and output will not be shown on the command window.
3. When there is no semicolon the duration was whereas when there was semicolon, the duration measured was which is shorter. Therefore, it is more useful to use semicolon when the output does not need appear on the command window.
4. a\*b = c is matrix multiplication operation. However, both a and b are 4x1 matrices and therefore MATLAB gives a dimension size error. The error:I  
   Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number ofrows in the second matrix. To operate on each element of the matrix individually, use TIMES (.\*) for elementwise multiplication.
5. “.\*” performs elementwise multiplication and multiplies a and b element by element and write it to matrix of same size in the same order of multiplication. a.\*b and b.\*a give the same result as multiplication is commutative.
6. This time since the matrix sizes are 1x4 and 4x1, MATLAB performs the operation and gives a 1x1 matrix as result of matrix multiplication. This is viable as the first one’s column size and second one’s row size is same. The result is 1x1 as the result will have first one’s row size and second one’s column size.
7. This type matrix multiplication gives of a 4x4 as the inputs are in the order of 4x1 and 1 x 4.
8. [starting point: step size :end point], This command essentially creates a row vector that has the entry of starting point and end of end point where in between there are elements seperated by the step size.
9. When there is a semicolon, it performs the operation faster than the one without the semicolon. MATLAB performed this operation in about seconds.
10. When a loop is created the duration increases to seconds
11. Allocating a memory shortens the duration as the place where the data gets saved is now ready for the MATLAB. The duration was about seconds.  
    In short, the duration comparison can be written as
12. Essentially Matlab performs an elementwise multiplication and creates an array on as the domain fo sin() function.
13. Plot (x) uses the elements of array as the x axis on plot.  
    Plot(x,t) uses x as x axis and plots t as y axis.  
    Plot(t,x) plots t on x axis and plots x as y axis.
14. Plot(t,x’-+’) connects the plot points and marks plot points with plusses whereas Plot(t,x’+’) draws the plot by only marking plot points with plusses

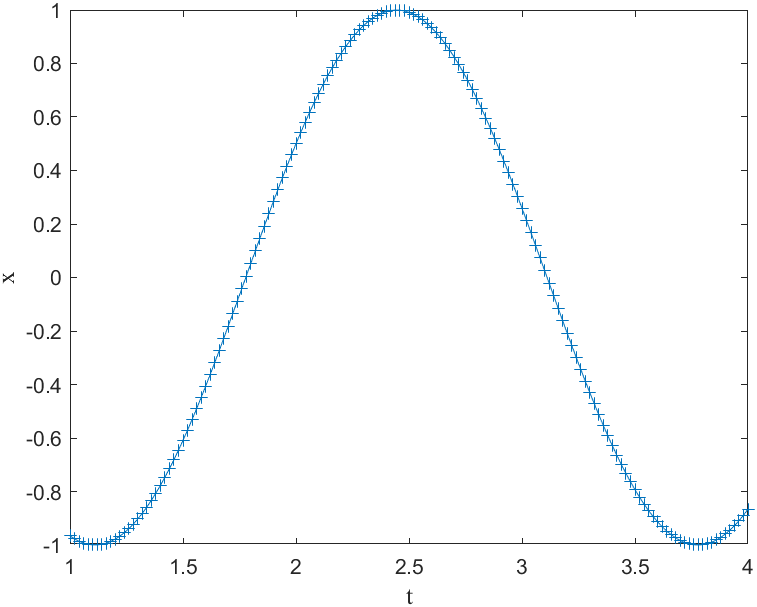
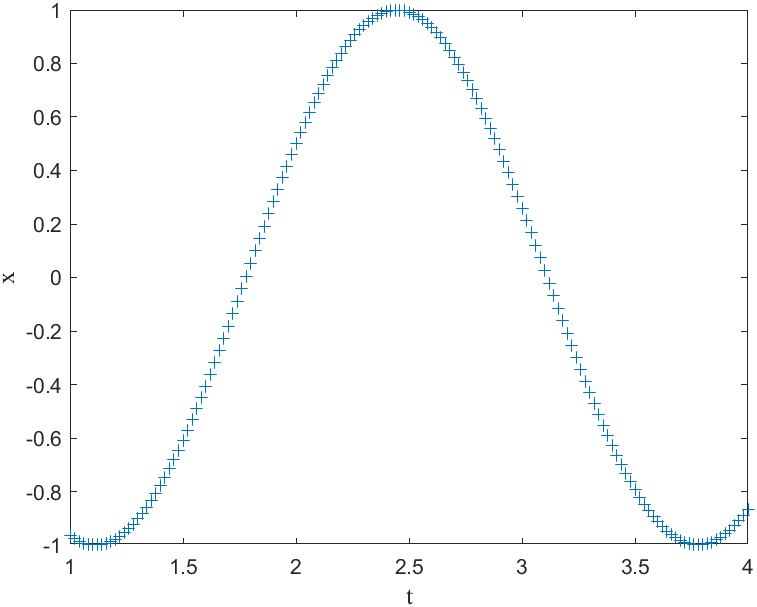
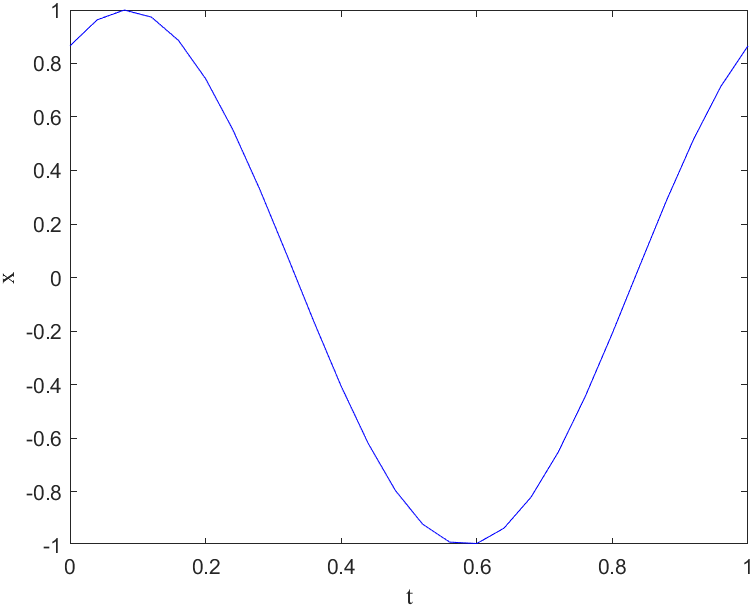


Figure 1&2: Plot(t,x,’+’) and Plot(t,x’-+’)

1. It can be calculated from which equals 26 points.
2. We need to find the amount of point which is,, 151 time points.   
   The linspace equivalent of this code is linspace is (1,4,151) .
3. x(t) can easily be seen by writing
4. Plot(t, x, ’b’) gives:

  
Figure 3: Plot(t, x, ’b’)

1. After using hold on the next plot will be made over the earlier plot

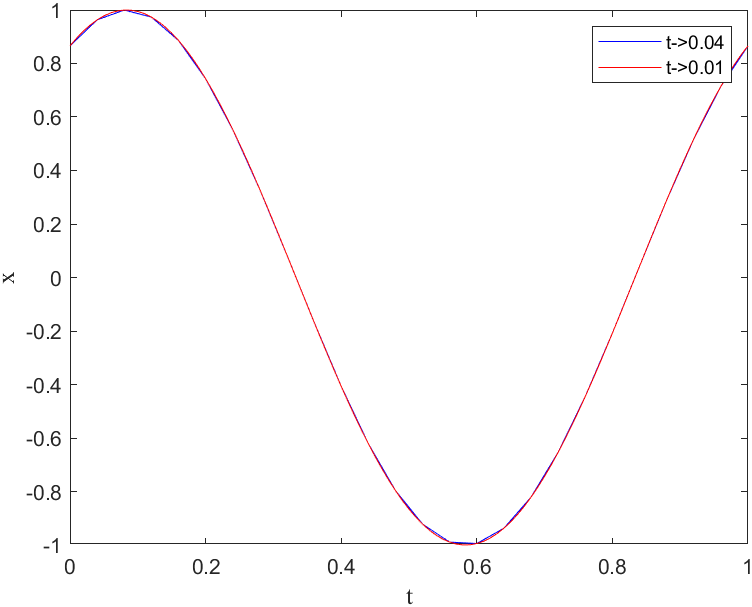


Figure 4: Higher step size(red) and slower step size (blue)

Though it is not exactly visible, on closeup it looks more detailed on the one with shorter stepsize as MATLAB gets more data to sinusoidal function.

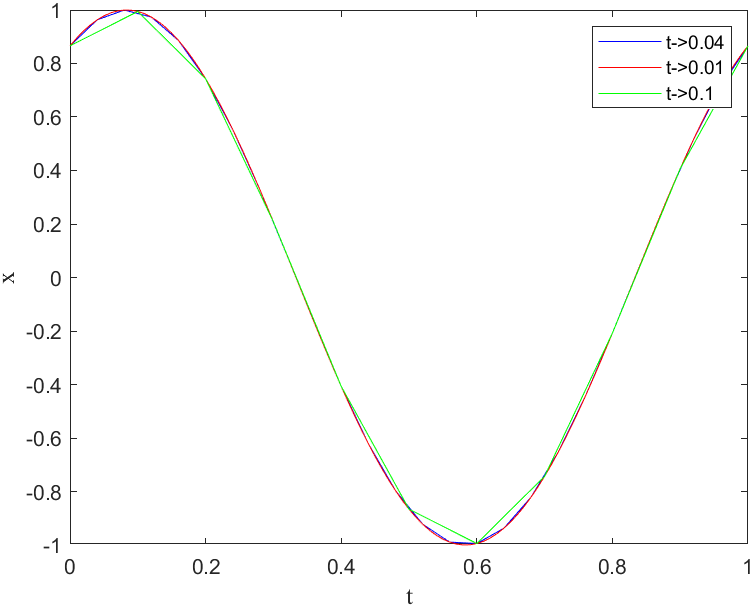


Figure 5: Step size 0.1 (green) addition

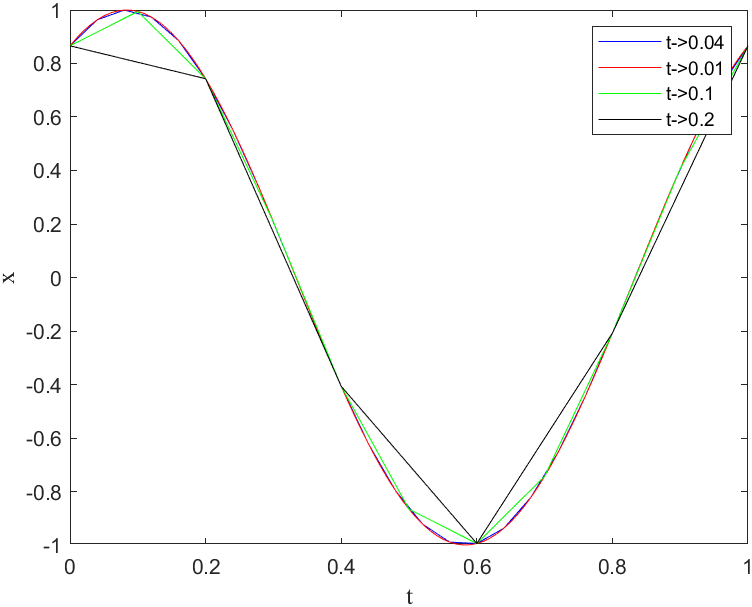


Figure 6: Step size 0.2 (black) addition

1. As step size gets higher there are less plot points for MATLAB to sample from the plot starts looking less like a sinusoidal wave. In such case the one with that provides most plot points, which would be t = 0:0.01:1, is closest to being continuous. As a continuous time function is defined for real numbers that makes up an infinite set. This requires the step size to be as small as possible to have as many sampling points as possible.

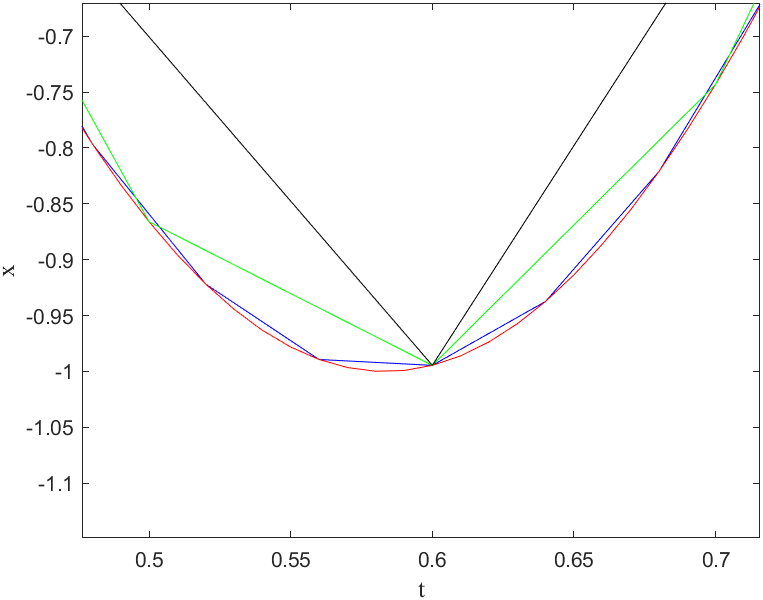


Figure 7: Task v

1. It connects the data points using lines.
2. Stem command can be used to create discrete data points as it does not connect data points. It represents data points individually.

Part 2

1. Sound() plays the sound at the sample rate of 8192Hz. Soundsc() scales the data so that it can be played as loud as possible at the default sample. Datas values are fit to in range of –1 to 1 when soundsc() is used. Since cos() function is being used both sound() and soundsc() can be used.
2. Sound() and soundsc() gives off the same sound.

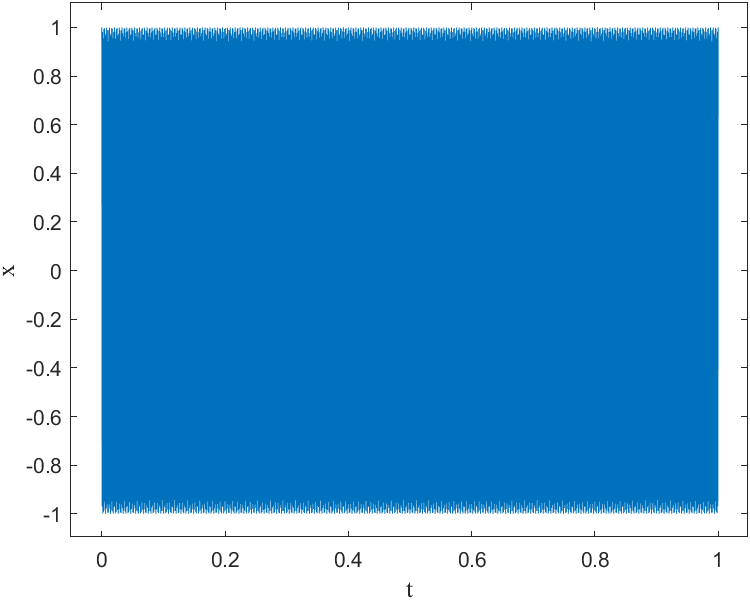


Figure 8: task b

c) and d) The sound’s pitch increases as the frequency increases.

Matlab code for elementwise part,

x= exp(-a\*t).\*cos(2\*pi\*f\*t);

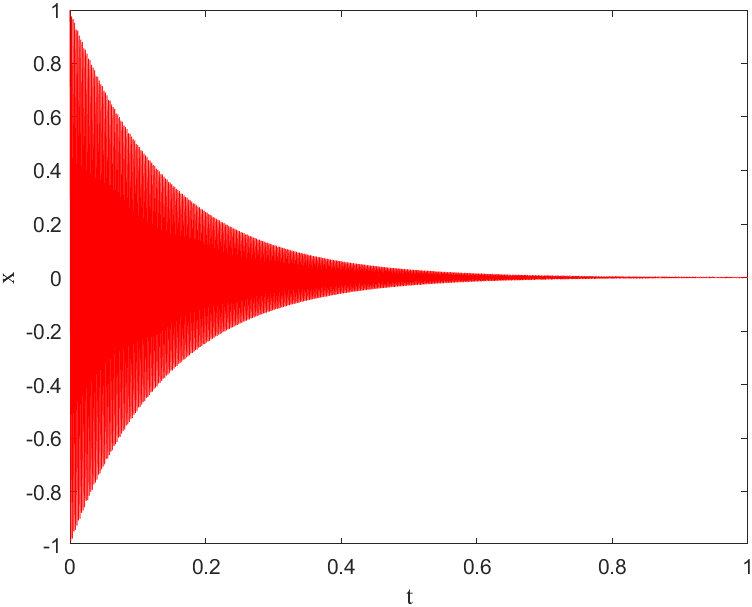
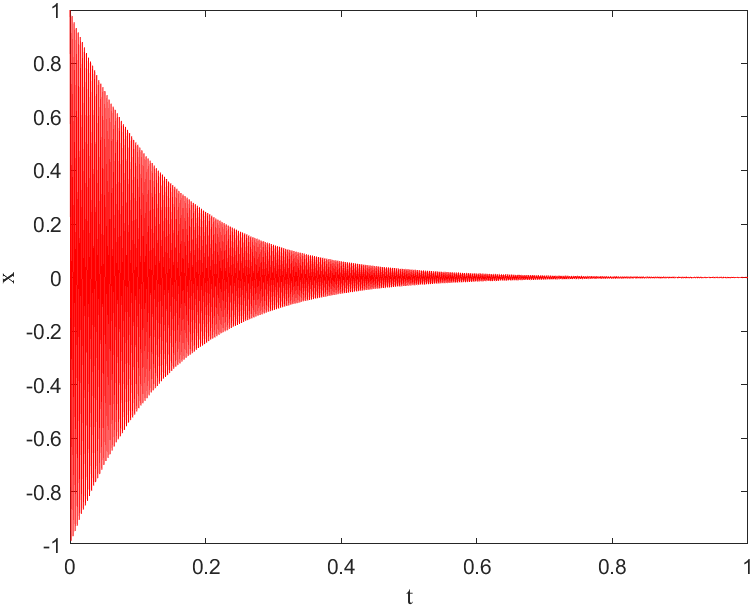


Figure 9 & 10: f0=330Hz and f0=440Hz

The pitch is higher when frequency is 440Hz and the exponential provides the sound to decrease exponentially causing x2 to have a pianolike sound. Whereas x1 as more flutelike noise as the amplitude does not change overtime.

When a’s value is changed to “2” the exponential decreases slower hence the sound seems to play longer, though it is the same duration. This is due to slower decrease in hearable frequency which lengthens the hearable duration.

When a’s value is changed to “11” the sound disappears faster as the hearable frequency is now much shorter.

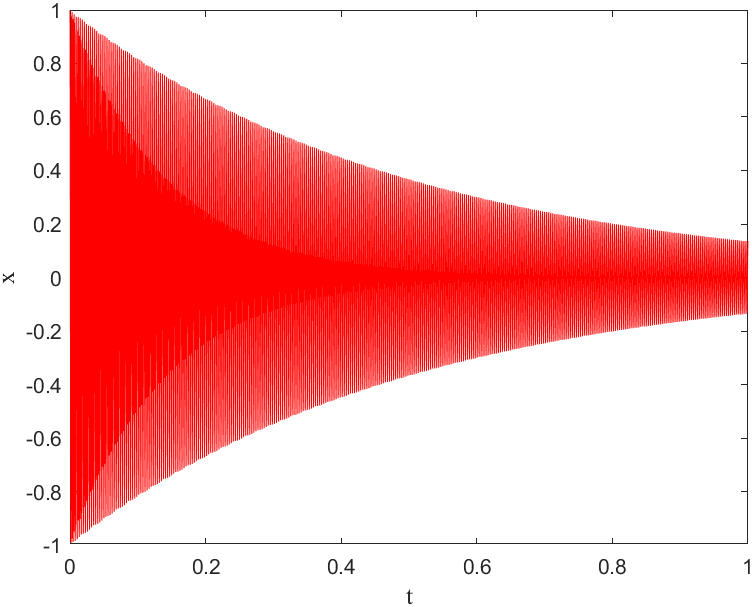


Figure 11: x= exp(-2\*t).\*cos(2\*pi\*330\*t)

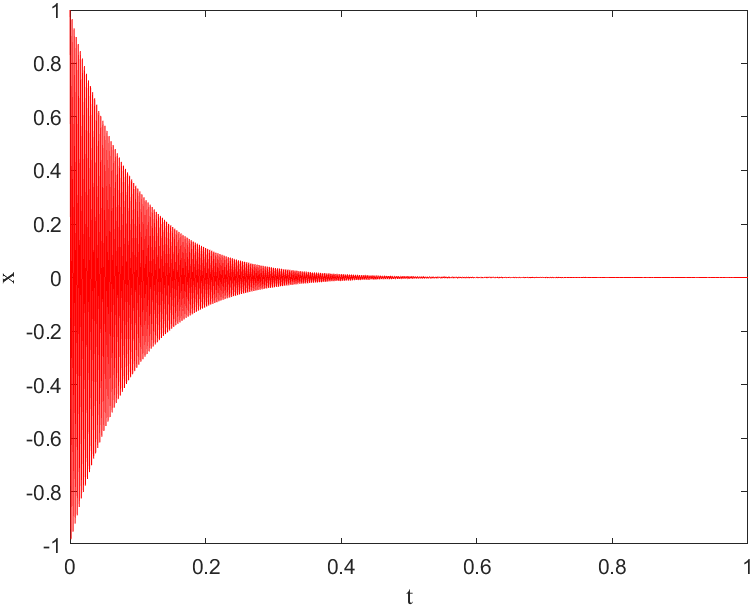


Figure 12: x= exp(-11\*t).\*cos(2\*pi\*330\*t)

Next section in part 2’s code:

x\_3= cos(2\*pi\*f1\*t).\*cos(2\*pi\*f0\*t);

Multiplying with a lower frequency creates an amplitude envelope which will change a sounds amplitude over time hence creating a sound that seems to be beating. If one increases the frequency of f1, the frequency of amplitude envelope will increase.

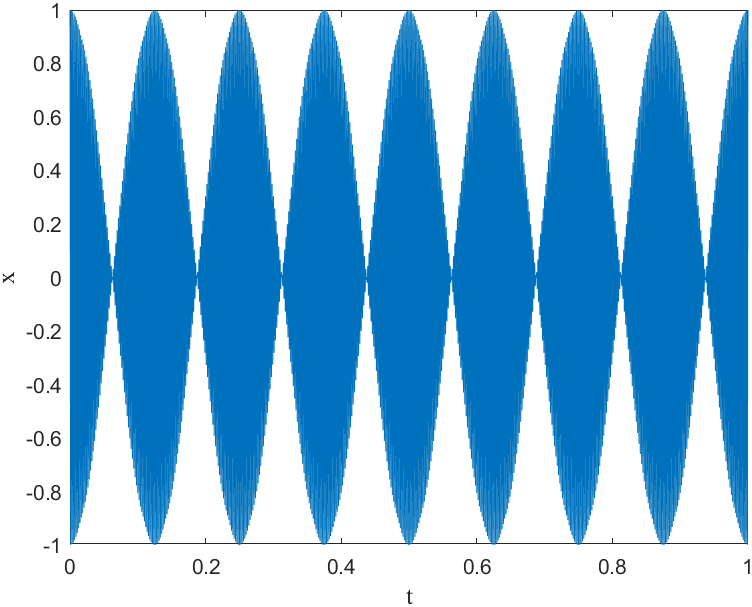


Figure 13: Amplitude Envelope (f1=4)

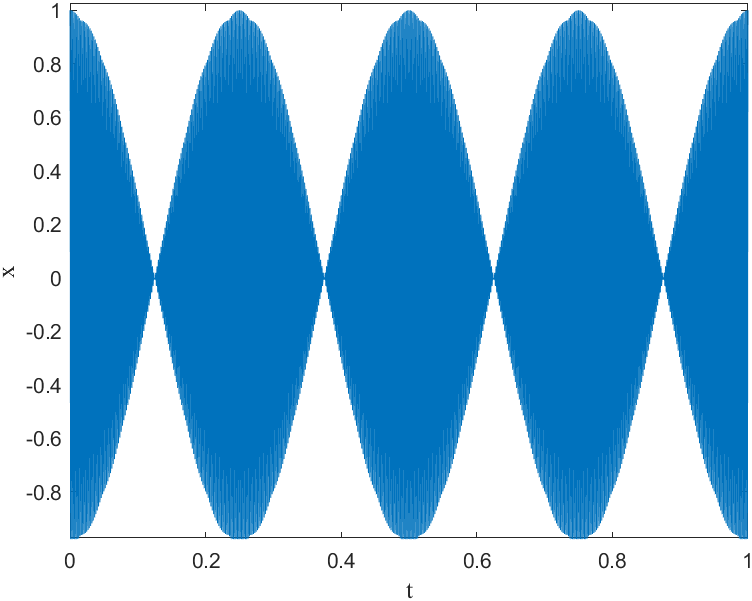


Figure 14: Amplitude Envelope (f1=2)

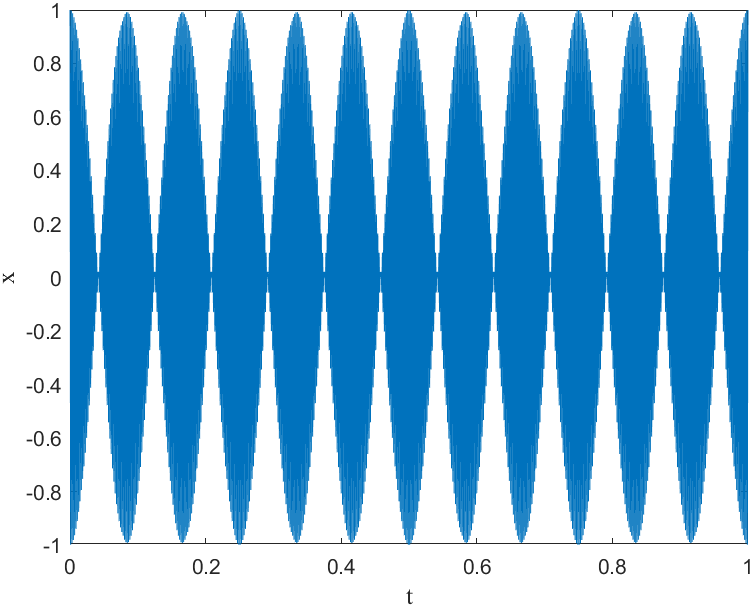
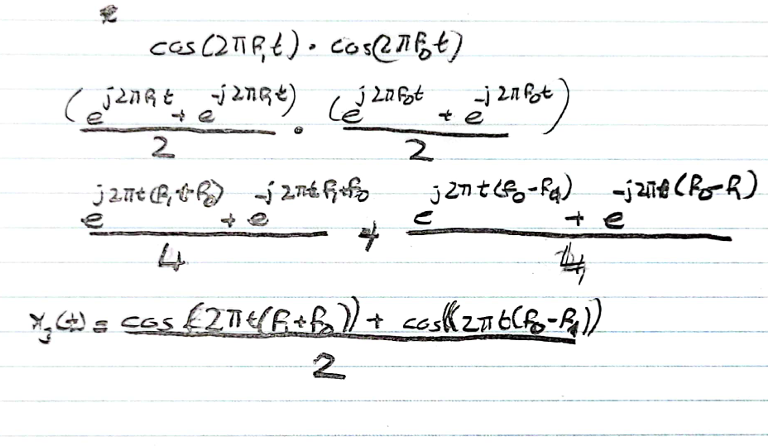


Figure 15: Amplitude Envelope(f1=6)

To be able to transform x\_3= cos(2\*pi\*f1\*t).\*cos(2\*pi\*f0\*t) into a sum of two different frequency cosine we can do:



The sound heard in this case is same.

Part 3

x1(t) =cos(2πf0t),

From this we can see that

,

Moving on to the part with x4, the code for x4 is:

x4 =cos(pi\*alpha\*t\*t);

The instantaneous frequency at t= 0 and t= t0 are:

The number random generator gives for is . Now x4 can be written as,

x4 =cos(pi\*alpha\*(t.^2));

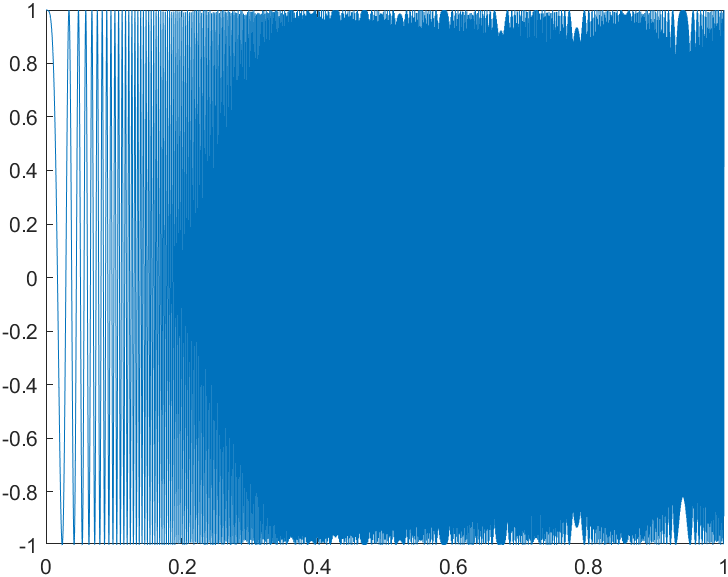


Figure 16:

As , The instantaneous frequency changes between 0 and 1744 hence why when a sound is played, the pitch seems to get higher exponentially. As increases, the frequency will increase at a faster rate as well.

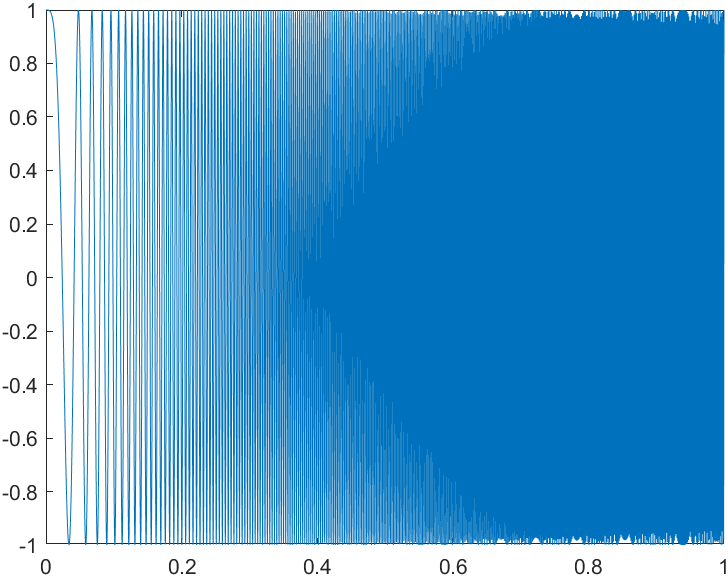


Figure 17:

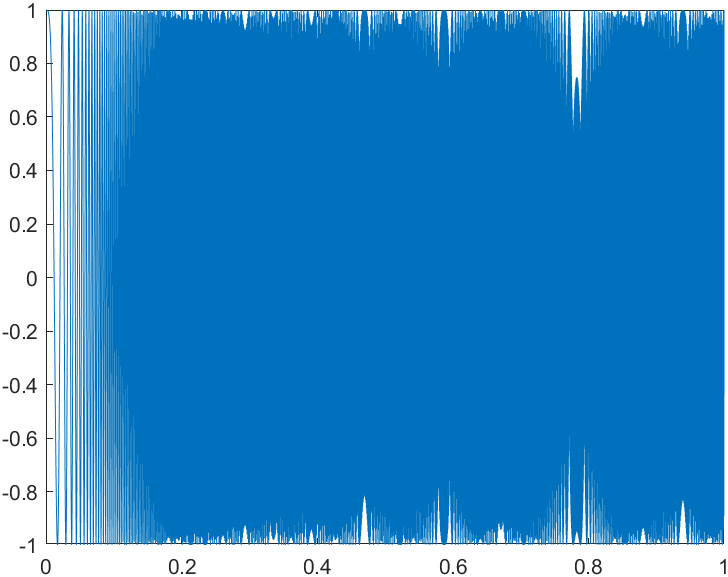


Figure 18:

Moving on to next section in part 3 we have:

x5(t) =cos(2π(−500t2+1600t)) whose code would be,

x5= cos(2\*pi\*(-500\*t.^2+1600\*t));

T is [0:1/8192:2].

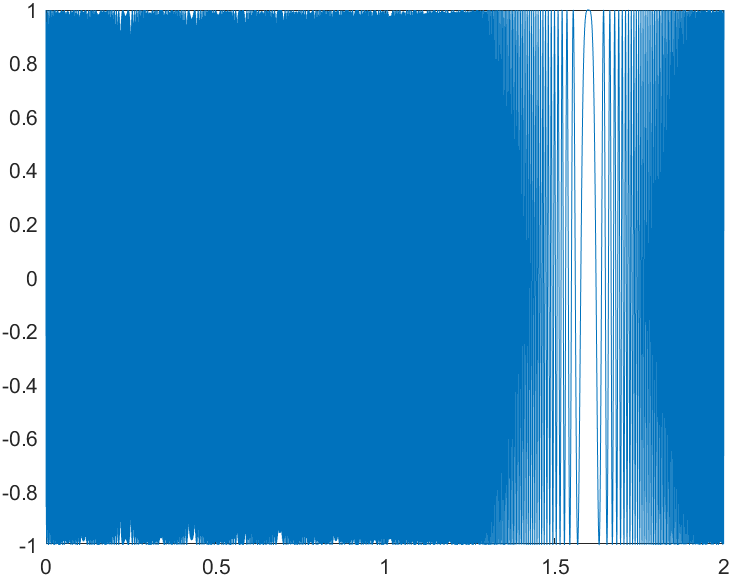


Figure 19:

Calculating the instantaneous frequency:

For t= 0,1 and 2 the corresponding instantaneous values are 1600Hz,600Hz and –400Hz in order. The minus sign only indicates the change and the actual frequency.

Part 4

We have,

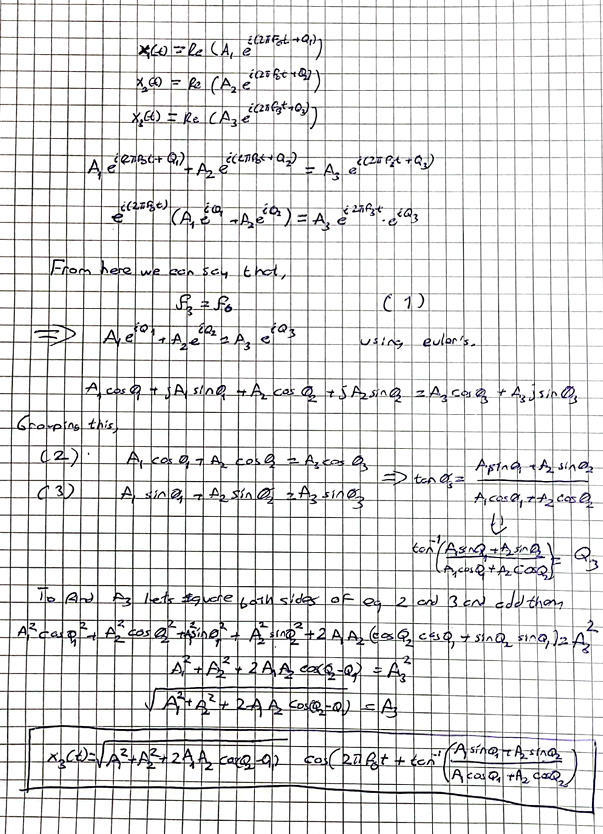
,

where from the last part.

The pitch of the sound does not change as the phase is changed as the change in phase does not effect the instantaneous frequency in anyway. To prove,

Part 5

To find x1(t)= A1cos(2πf0t+φ1) and x2(t)= A2cos(2πf0t+φ2)’s addition as   
x3(t) = A3cos(2πf3t +φ3):



A3 is maximum when that is when

A3 is minimum when that is when

On max value

On min value