Bilkent University

EEE321: Signals and Systems

Lab Assignment 1

Mehmet Emre Uncu - 22003884

Section - 01

Part 1:

a) When a = [3.2 34/7 -6 24] and a = [3.2; 34/7; -6; 24] were typed, the first expression was interpreted as a row vector, but since there was a semicolon between the values, the second expression was interpreted as a column vector.

b) When a semicolon was added to the end of the expressions, unlike part a, the output was not displayed in the command window.

c) When a semicolon was not used, the elapsed time was 0.000127 seconds; when a semicolon was used, the elapsed time was 0.000046 seconds. So the time difference is 0.000081 seconds. Because of this time saving, using semicolons where the output does not need to be displayed efficiently reduces elapsed time.

d) The message is:

> *Error using **

> *Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in the second matrix. To operate on each matrix element individually, use TIMES (.*) for elementwise multiplication.*

The a * b operation is used for matrix multiplication. Still, in this example, the number of columns in the first matrix does not match the number of rows in the second matrix, so a and b are unsuitable for this operation.

e) The result is:

c =

*1.0e+03 **

0.0186 0.0233 -0.0300 -2.4480

When `.*` was used instead of `*`, the operation changed to elementwise multiplication. When c = b.*a was written instead of c = a.*b, the result did not change because multiplication is commutative.

f) The result is a 1x1 matrix:

c =

-2.4361e+03

The number of columns in the first matrix matches the number of rows in the second matrix, so a(4x1) and b(1x4) are now suitable for this operation, and MATLAB performed a matrix multiplication.

g) The result is a 4x4 matrix:

c =

*1.0e+03 **

0.0186 0.0154 0.0160 -0.3264
0.0282 0.0233 0.0243 -0.4954
-0.0348 -0.0288 -0.0300 0.6120
0.1392 0.1152 0.1200 -2.4480

The number of columns in the first matrix matches the number of rows in the second matrix, so a(1x4) and b(4x1) are now suitable for this operation, and MATLAB performed a matrix multiplication.

h) The command creates a row vector `a` containing values from 1 to 2 in increments of 0.01.

i) The elapsed time is 0.000093 seconds.

j) The command is:

```
a = [];
for i = 1:100
    a = [a, 1 + (i - 1) * 0.01];
end
```

The elapsed time is 0.000314 seconds.

k) The command is:

```
a = zeros(1, 100);
for i = 1:100
    a(i) = 1 + (i - 1) * 0.01;
end
```

The elapsed time is 0.000088 seconds.

The method in part k is the most efficient because it preallocates memory for the entire vector before filling it. The other ways are less efficient because dynamic array resizing can be slower as the array grows.

l) MATLAB performed elementwise operations, put each element in vector a separately into the function, and stored the results in b.

m) plot(x) uses the array indices as the x-axis.

plot(x,t) uses the x array as the x-axis and the t array as the y-axis.

plot(t,x) uses the t array as the x-axis and x array as the y-axis.

n) The first command plots '+' signs on the data points and connects them with the line. The second command plots '+' signs on the data point only.

Figure 1: plot(t,x,'-+') and plot(t,x,'+')

o) By using ((1 – 0) / 0.04) + 1, t has 26 time points.

p) The t in part m has ((4-1) / 0.02) + 1 = 151 time points. To generate the variable t in part m, the command should be t = linspace(1,4,151).

q) ok

r)

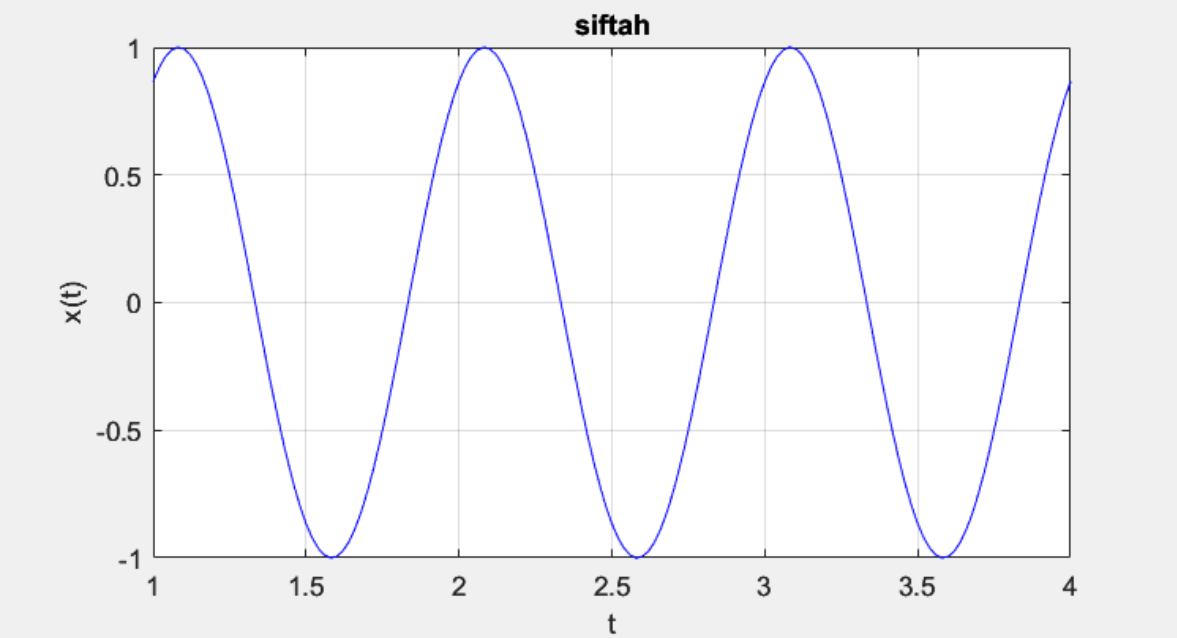

Figure 2: The plot of x(t)

s) By using ((1 – 0) / 0.01) + 1, t has 101 time points.

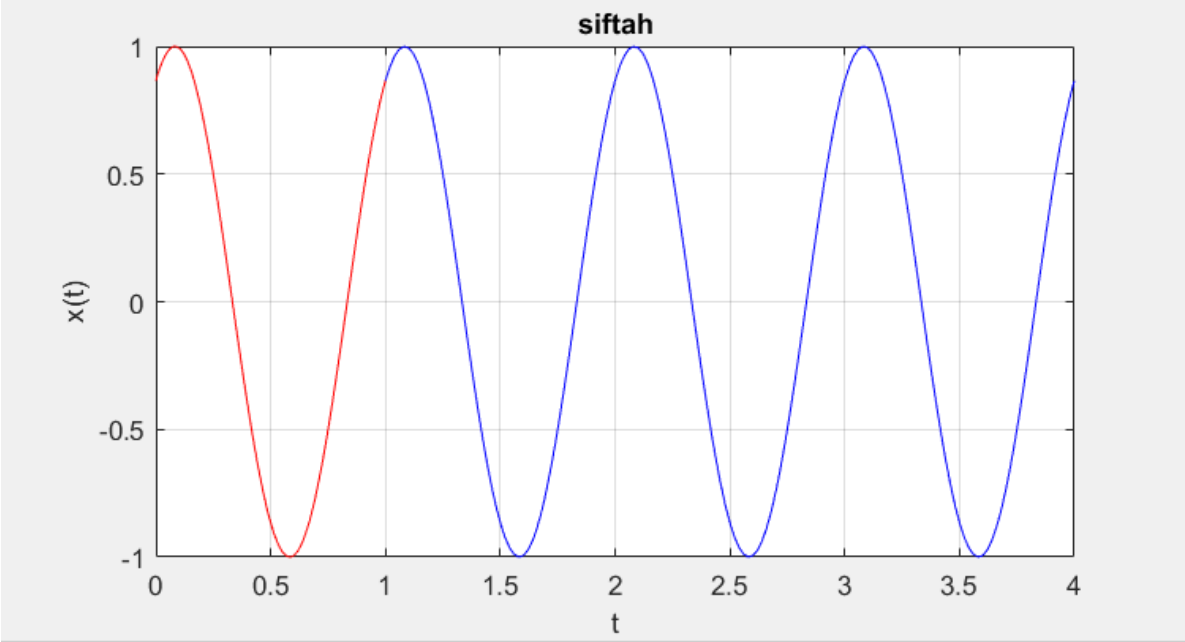

Figure 3: The plot of x(t)

t) By using ((1 – 0) / 0.1) + 1, t has 11 time points.

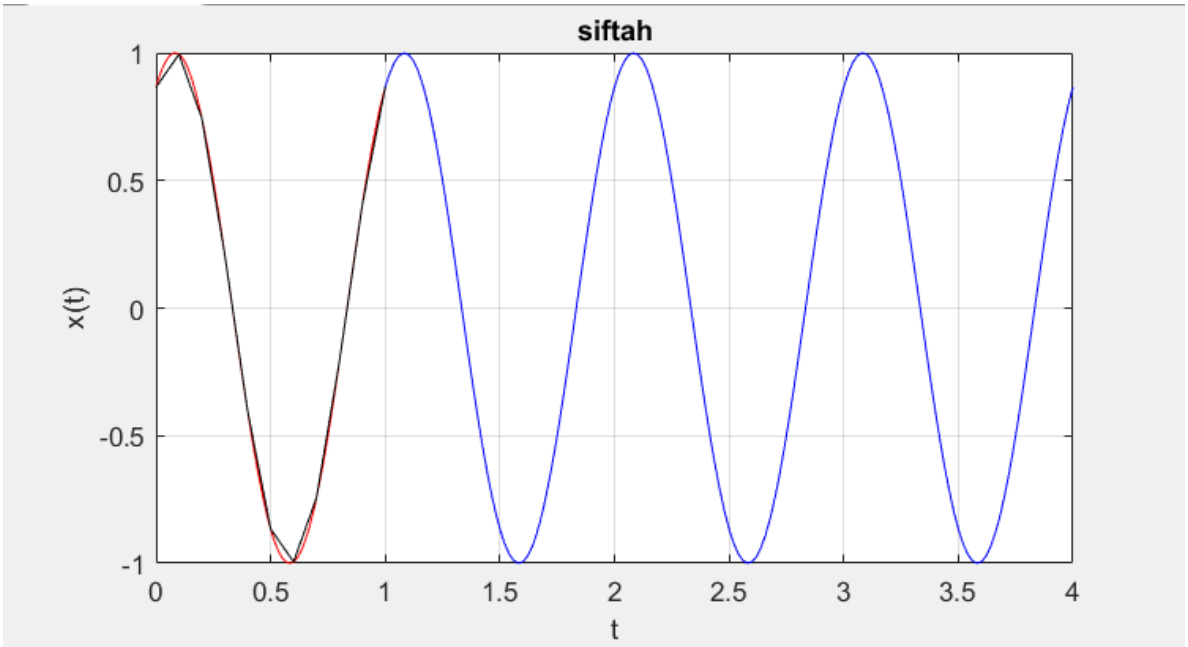

Figure 4: The plot of x(t)

u) By using ((1 – 0) / 0.2) + 1, t has 6 time points.

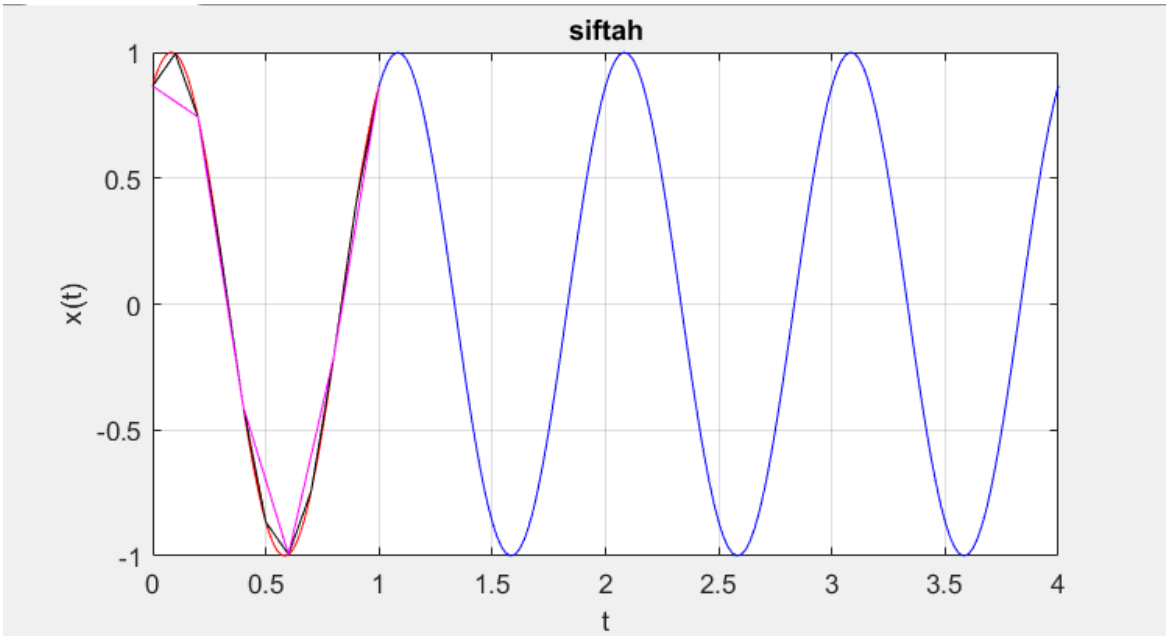

Figure 5: The plot of x(t)

v) The best choice is the graphic in part r (blue) because it has the most time points in t. As the number of time points increases, as the difference between the data points decreases, it becomes possible to observe the behavior of continuous x(t) more accurately.

w) The plot command connects the data points with straight lines.

x) The plot command connects the data points and creates line plots for continuous data. The stem command represents data points with vertical lines instead of connecting them and creates a stem plot for discrete data.

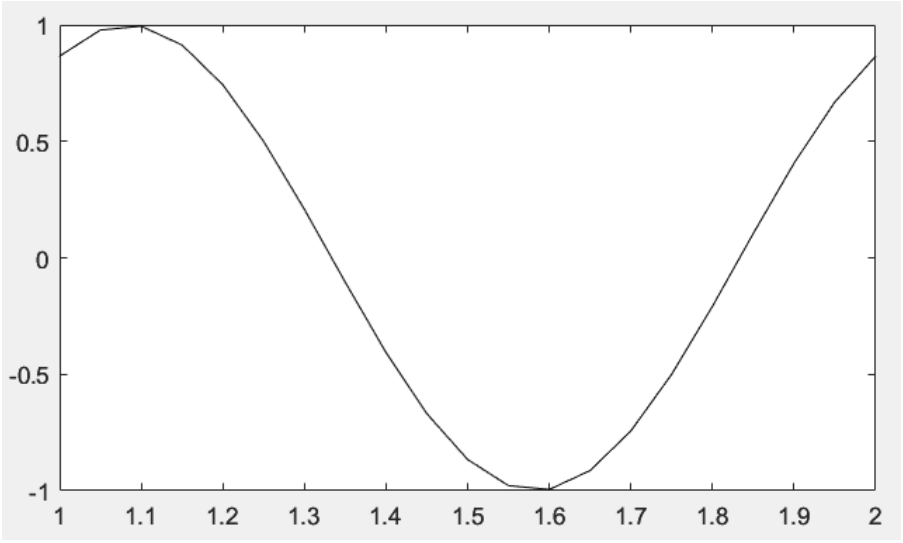

Figure 6: plot command

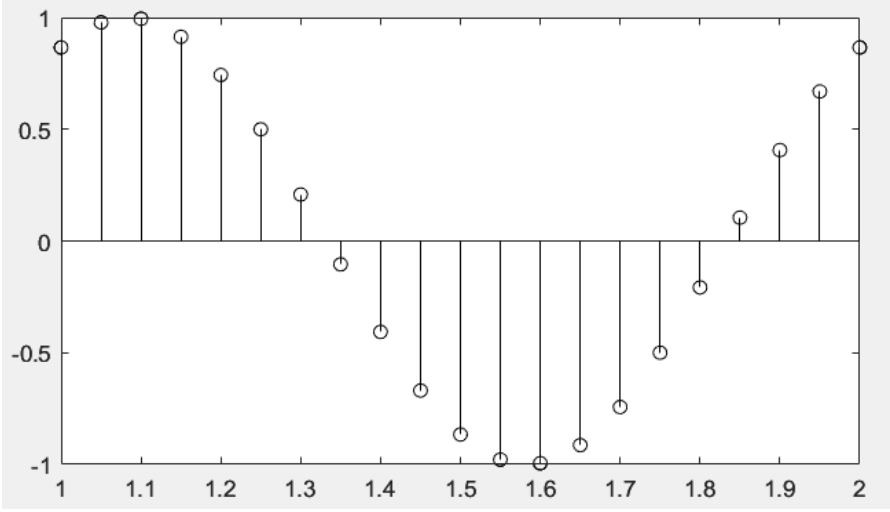

Figure 7: stem command

Part 2:

a) The soundsc() command scales the value of the signal to fit in the range between -1 to 1, then sends the data, but the sound() command sends the default data directly. When the sound() command is used, the signal can be clipped according to the limit of the sound card. But such a situation does not occur with the soundsc() command. Since the cos signal is currently between -1 and 1, it is appropriate to use both commands.

b) The sound does not change when the sound() or soundsc() is used because of the range of cos.

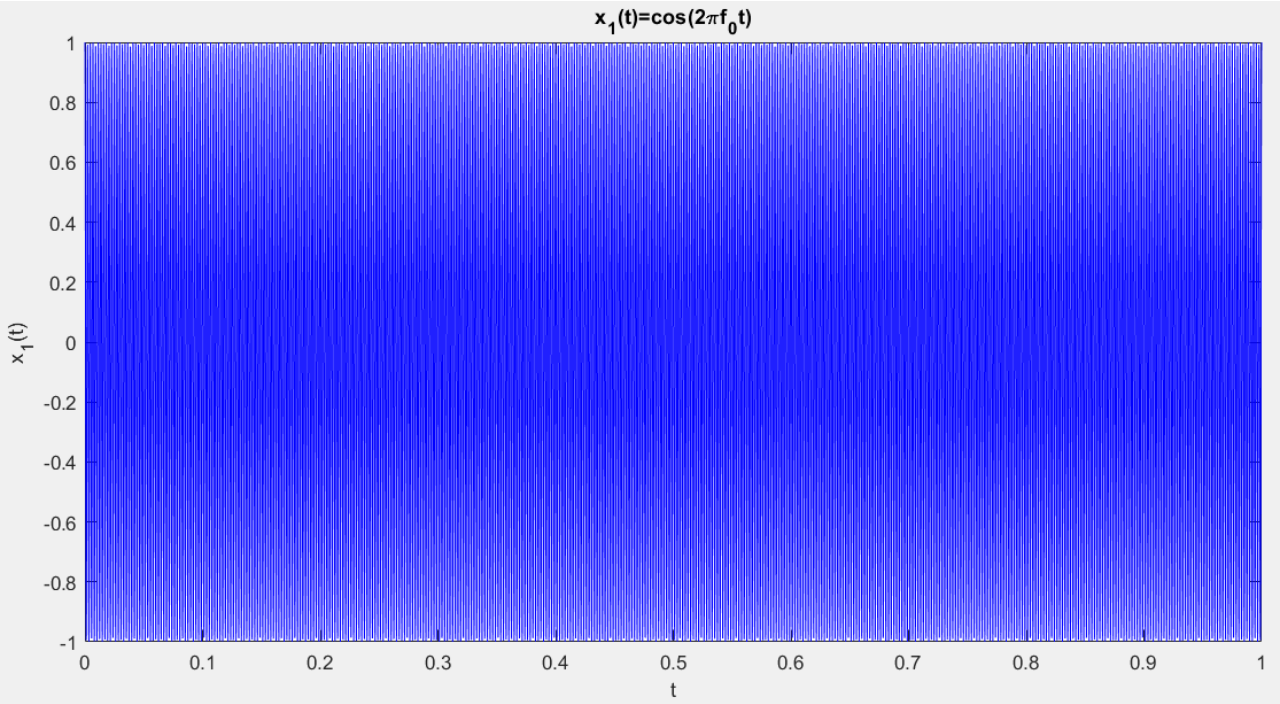

Figure 8: part 2.b

c) The test was repeated with a higher frequency.

d) The test was repeated with a higher frequency, and it was observed that as the frequency increased, the pitch of the sound increased.

e) MATLAB code:

```
t  = 0:1/8192:1;
f0 = 330;
a  = 7;
x2 = exp(-a*t).*cos(2*pi*f0*t);
plot(t,x2);
title('x_2(t)=e^{-at}cos(2\pif_0t)');
sound(x2);
```

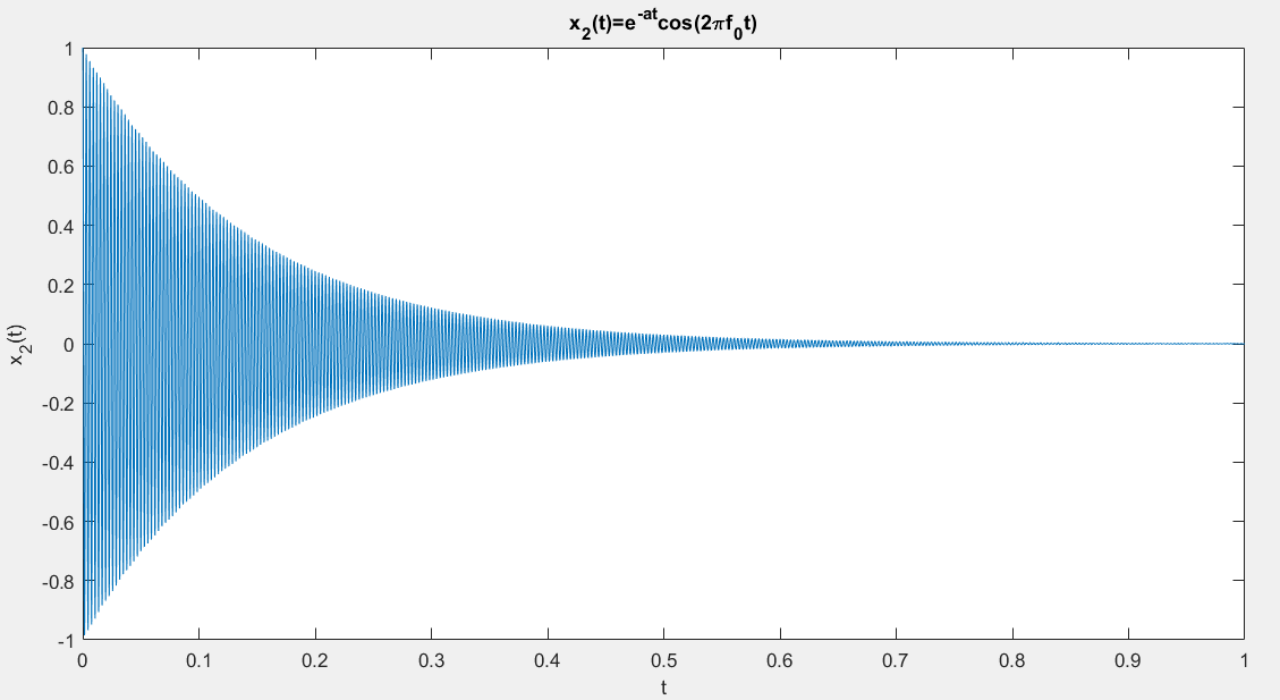

Figure 9: plot of the signal with a = 7

When e^{-at} was added to the $x_1(t)$ signal, a gradually decaying oscillation occurred, as seen in the plot. While the sound of $x_1(t)$ remained constant, the sound of $x_2(t)$ gradually became lower and eventually diminished. Due to its exponentially decreasing amplitude, the sound of $x_2(t)$ resembles a piano, while the constant sound of $x_1(t)$ resembles a flute.

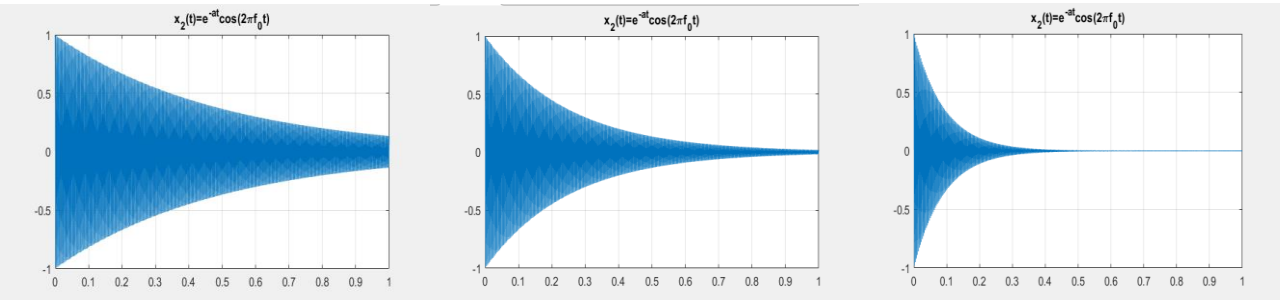

Figure 10: plot of the signal with a = 2,4 and 11

Increasing 'a' decreases the duration of the sound because it makes the amplitude decay faster. Thus, as 'a' increased, the sound became quieter more quickly.

f) MATLAB code:

```
t  = 0:1/8192:1;
f0 = 510;
f1 = 4;
x3 = cos(2*pi*f1*t).*cos(2*pi*f0*t);
plot(t,x3);
title('x_3(t)=cos(2\pif_1t)cos(2\pif_0t)');
sound(x3);
```

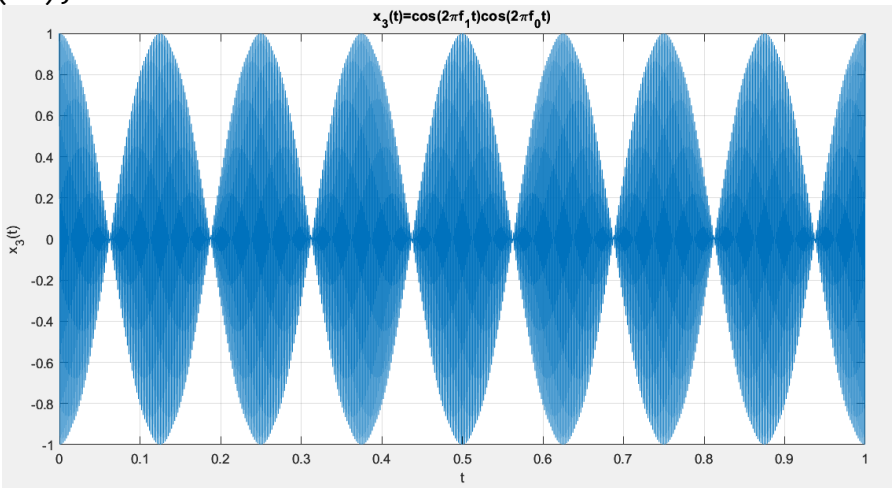

Figure 11: plot of the signal with $f_1 = 4$

The low-frequency cosine term forms an envelope added to $x_1(t)$, creating a sound that repeatedly becomes louder and quieter. Thus, the $x_1(t)$ signal, which oscillates constantly, turned into a signal that oscillates with the period of the low-frequency cosine term.

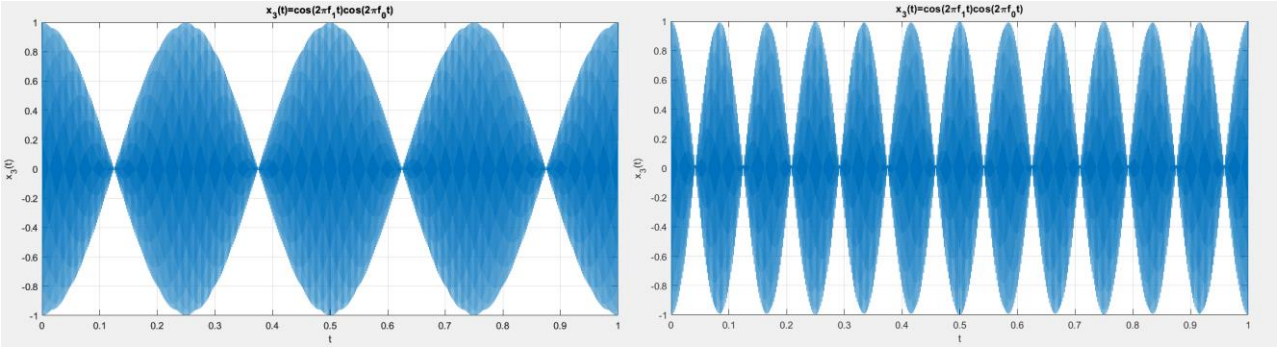

Figure 12: plot of the signal with $f_1 = 2$ *and 6*

When $f_1 = 2$, it became quieter and louder slower; when $f_1 = 6$, it became quieter and louder faster. So, the time it takes for the sound to go from maximum to minimum has changed.

$$x_3(t) = \cos(2\pi f_1 t)\cos(2\pi f_0 t) = \frac{1}{2}[\cos(2\pi t(f_0 + f_1)) + \cos(2\pi t(f_0 - f_1))]$$

This way, x3 can be written as the sum of two cosines. The sound of $x_3(t)$ can interpreted as the sound of two cosines with different frequencies played on top of each other.

Part 3:

The instantaneous frequency $f_{ins}(t) = \frac{d\phi(t)}{dt}$ $for\ x(t) = cos(2\pi\phi(t))$

To find the instantaneous frequency of the signal $x_1(t) = cos(2\pi f_0 t)$:

$$f_{ins} = \frac{d}{dt}(f_0 t) = f_0$$

So it can be seen that $f_{ins} = f_0$ for all t.

To find the instantaneous frequency of the signal $x_4(t) = cos(\pi\alpha t^2)$:

$$f_{ins} = \frac{d}{dt}\left(\frac{\alpha t^2}{2}\right) = \alpha t$$

So it can be seen that $f_{ins} = \alpha t$ for all t. When t = 0, $f_{ins} = 0$ and at t = t_0, $f_{ins} = \alpha t_0$

The randomly selected 'a' is 1984. The range of f_{ins} is from 0 to 1984, and the frequency values will change between this range.

MATLAB code:

```
t  = 0:1/8192:1;
a = 1984;
x4 = cos(pi*a*t.^2);
plot(t,x4);
title('x_4(t)=cos(\piat^2)');
sound(x4);
```

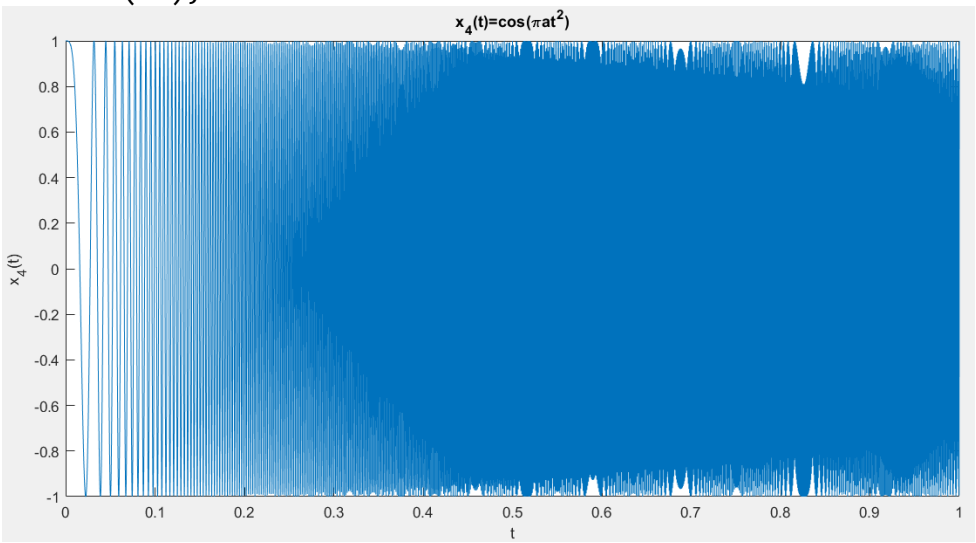

Figure 13: plot of the signal with a = 1984

The signal starts from a low pitch and gets higher as time passes. That is because the instantaneous frequency increases linearly until 1984Hz.

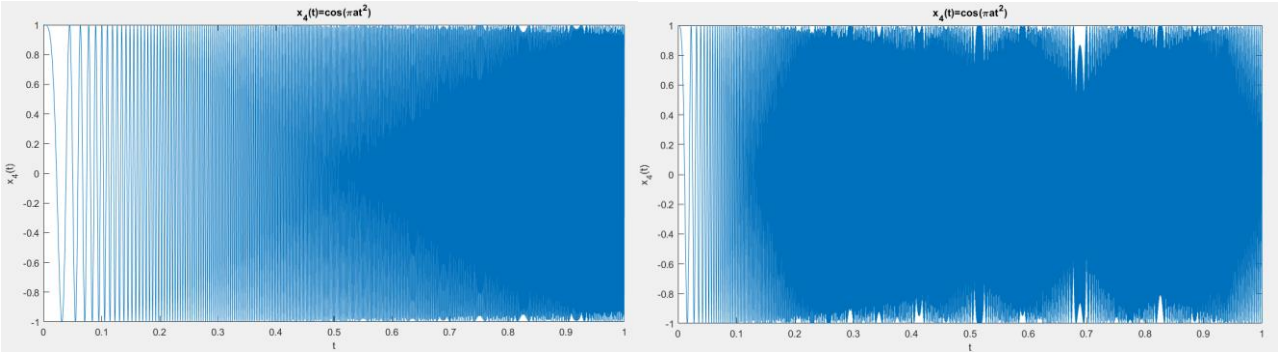

*Figure 14: plot of the signal with a = 1984/2 and a = 1984*2*

For a = 1984/2, the max frequency value is 992Hz, and its sound becomes less high-pitched than the previous signal. For a = 1984*2, the max frequency value is 3968Hz, and the sound becomes higher pitched than the previous signal.

MATLAB code:

```
t  = 0:1/8192:2;
x5 = cos(2*pi*(-500*t.^2+1600*t));
plot(t,x5);
title('x_5(t)=cos(2\pi(-500t^2+1600t))');
sound(x5);
```

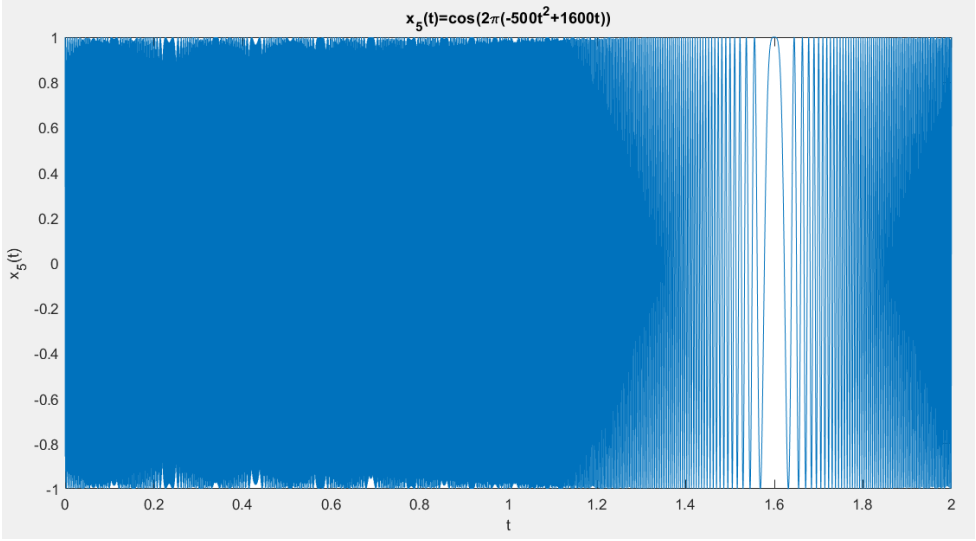

Figure 15: plot of the $x_5(t)$ signal

The frequency of the signal starts high and gradually gets lower.

To find the instantaneous frequency of the signal $x_5(t) = \cos(2\pi(-500t^2 + 1600t))$:

$$f_{ins} = \frac{d}{dt}(-500t^2 + 1600t) = -1000t + 1600$$

$$f_{ins}(0) = 1600, \quad f_{ins}(1) = 600, \quad f_{ins} = -400$$

Part 4:

MATLAB code:

```
t  = 0:1/8192:1;
a = 1984;
p = 0;
x = cos(2*pi*a*t+p);
plot(t,x);
title('x(t)=cos(2\piat+p)');
sound(x);
```

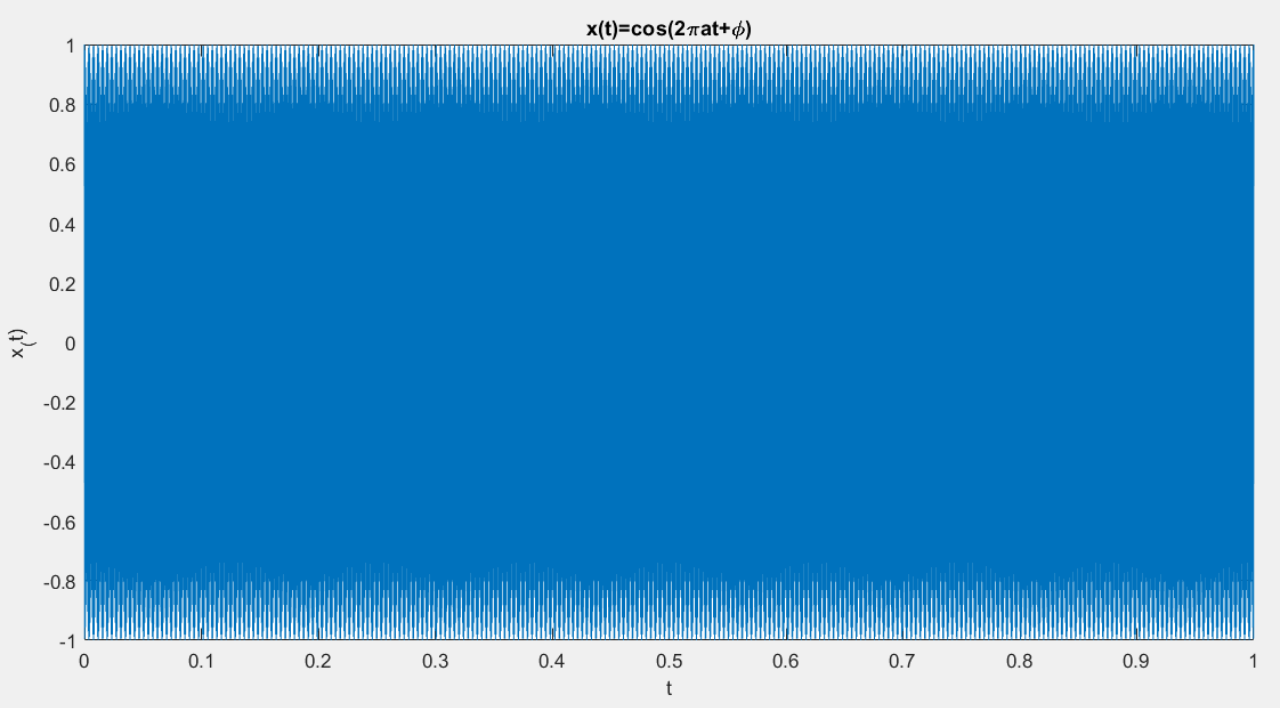

Figure 16: plot of the signal when ϕ = 0

Changing the value of ϕ does not affect the signal's pitch or volume. The change in ϕ only changes the initial phase of the generated cosine wave.

Part 5:

$$x_1(t) = A_1\cos(2\pi f_0 t + \varphi_1)$$

$$x_2(t) = A_2\cos(2\pi f_0 t + \varphi_2)$$

$$x_3(t) = A_3\cos(2\pi f_3 t + \varphi_3)$$

$$x_1 + x_2 = x_3$$

$$x_1 + x_2 = [A_1\cos(2\pi f_0 t)\cos(\varphi_1) - A_1\sin(2\pi f_0 t)\sin(\varphi_1)] +$$
$$[A_2\cos(2\pi f_0 t)\cos(\varphi_2) - A_2\sin(2\pi f_0 t)\sin(\varphi_2)]$$

$$x_1 + x_2 = [A_1\cos(\varphi_1) + A_2\cos(\varphi_2)]\cos(2\pi f_0 t) - [A_1\sin(\varphi_1) + A_2\sin(\varphi_2)]\sin(2\pi f_0 t)$$
$$x_3 = [A_3\cos(\varphi_3)]\cos(2\pi f_3 t) - [A_3\sin(\varphi_3)]\sin(2\pi f_3 t)$$

The sum should be the same frequency with two cosines, $f_3 = f_0$

$$A_1\cos(\varphi_1) + A_2\cos(\varphi_2) = A_3\cos(\varphi_3)$$
$$A_1\sin(\varphi_1) + A_2\sin(\varphi_2) = A_3\sin(\varphi_3)$$

$$\tan(\varphi_3) = \left(\frac{[A_1\sin(\varphi_1) + A_2\sin(\varphi_2)]}{[A_1\cos(\varphi_1) + A_2\cos(\varphi_2)]}\right)$$
$$\varphi_3 = \arctan\left(\frac{[A_1\sin(\varphi_1) + A_2\sin(\varphi_2)]}{[A_1\cos(\varphi_1) + A_2\cos(\varphi_2)]}\right)$$

$${A_1}^2 + {A_2}^2 + 2A_1A_2\cos(\varphi_1 - \varphi_2) = {A_3}^2$$

$$A_3 = \sqrt{{A_1}^2 + {A_2}^2 + 2A_1A_2\cos(\varphi_1 - \varphi_2)}$$

$$For\ k\ \in\ \mathbb{Z}$$

If $\varphi_1 - \varphi_2 = (2k-1)\pi,\ \cos(\varphi_1 - \varphi_2) = -1\ and\ A_{3_{min}} = \sqrt{{A_1}^2 + {A_2}^2 - 2A_1A_2} = A_1 - A_2$

If $\varphi_1 - \varphi_2 = 2k\pi,\ \cos(\varphi_1 - \varphi_2) = 1\ and\ A_{3_{max}} = \sqrt{{A_1}^2 + {A_2}^2 + 2A_1A_2} = A_1 + A_2$