EEE 321-Signals and Systems

-Lab 01-

Defne Yaz Kılıç
Section 001
22102167
26.09.23

Part 1

a) First type a=[3.2 34/7 -6 24], then type a=[3.2; 34/7; -6; 24]. What is the difference?

When we use the space between the elements Matlab interprets as four elements in a single row e.g. 1x4 row matrix for a=[3.2 34/7 -6 24], however if we use semicolons they are interpreted as elements of a 4x1 column matrix for a=[3.2; 34/7; -6; 24].

b) Type a=[3.2 34/7 -6 24]; and b=[3.2; 34/7; -6; 24];. What is the difference from part a?

In this case, the statements are finalized by a semicolon which suppresses the display output in the command window. In part a, when we run the commands the variable values are automatically displayed in the command window.

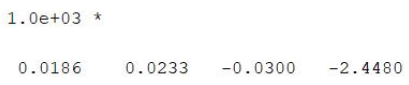
c) Time measurement with tic and toc commands

Without the semicolon as explained in the previous question, Matlab displays the variable in the command window hence the display time has also an effect on the execution time of the code. Without the semicolon the execution time of the variable is 0.000880 seconds. However when the semicolon is used, the variable is not displayed in the command window and the execution time is therefore less than the displayed case. With the semicolon the execution time is 0.000089 seconds. The time difference between without and with semicolon is Δt= 0.000791 seconds. It is useful to put semicolons when the output does not need to be displayed and when the time efficiency is crucial.

d) Matrix multiplication

An error message occurs saying "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 element of the matrix individually, use TIMES (.*) for element-wise multiplication." Since the elements of both matrices are separated with space, Matlab executes them as two 1x4 row matrices. Hence when we try to multiply them with "*" command it gives an error. For a correct matrix multiplication the number of columns of the first matrix should be equal to the number of rows of the second.

e) The dot in front of the multiplication symbol is used for element-wise multiplication, each column of the first matrix is pairwise multiplied with the second matrix's columns resulting a new 1x4 matrix. Since the element-wise multiplication is commutative the order of the matrices does not matter. **c=a.*b** and **c=b.*a** are equal to each other. The result is

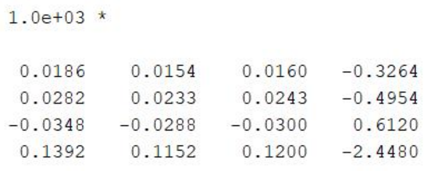


```
  1.0e+03 *

    0.0186    0.0233   -0.0300   -2.4480
```

f) a=[3.2 34/7 -6 24] is a 1x4 row matrix and b=[5.8; 24/5; 5; -102] is a 4x1 column matrix, when we type the c=a*b command Matlab executes matrix multiplication(axb). Multiplication of 1x4 and 4x1 matrices results in a 1x1 matrix : [-2.4361e+03].

g) This time the matrices are reversed, that is a=[3.2; 34/7; -6; 24] is now a 4x1 column matrix and b=[5.8 24/5 5 -102] is a 1x4 row matrix. The c=a*b command is now results in a 4x4 matrix:



```
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
```

h) The command a=[1:0.01:2] creates a 1x101 row matrix with 1 as the first column and by incrementing the previous columns by 0.01 the other columns are formed until 2 as the last column. In other words, the command divides the interval [1,2] into pieces with 0.01 step size and stores the data in matrix form.

i) The time it takes to create the matrix stated above using the command a=[1:0.01:2], takes 0.000079 seconds.

j) Using a for loop the generation time becomes 0.000790 seconds.

k) By allocating the memory of matrix variable a using a=zeros(1,101); command the generation time becomes 0.000446 seconds. When we compare the generation times of three different methods we see that $t_i < t_k < t_j$. The Matlab built in function is the most efficient one for creating such arrays. Comparing the two for loop methods we can see that the memory allocation reduces the generation time.

l) The sin function in Matlab operates element-wise on arrays. Hence the sin function takes the input [0:pi/8:2*pi] and gives a new array by applying the sin function to each element of the input.

m) plot(x): Plots the columns of x versus their index,it uses array x as the Y axis.
plot(x,t): Plots the columns of t versus the columns of x, the array x is the x axis and the array t is the y axis.
plot(t,x): Plots the columns of x versus the columns of t, the array t is the x axis an the x is the x axis.

n) plot(t,x,“-+”) marks the data points with an + over the continuous line; plot(t,x,“+”) only marks the data points the graph is discrete.

o) Using the equation [(1.0 - 0.0)/0.04]+1.0 ; 26 time points are included in t=[0:0.04:1].

p) In part m; from the same relation above t=[1:0.02:4] contains 151 time points [(4-1)/0.02]+1 = 151. Hence we can use t= linspace (1,4,151) command to generate the same variable in part m.

q) The graph of x(t) can be seen below:

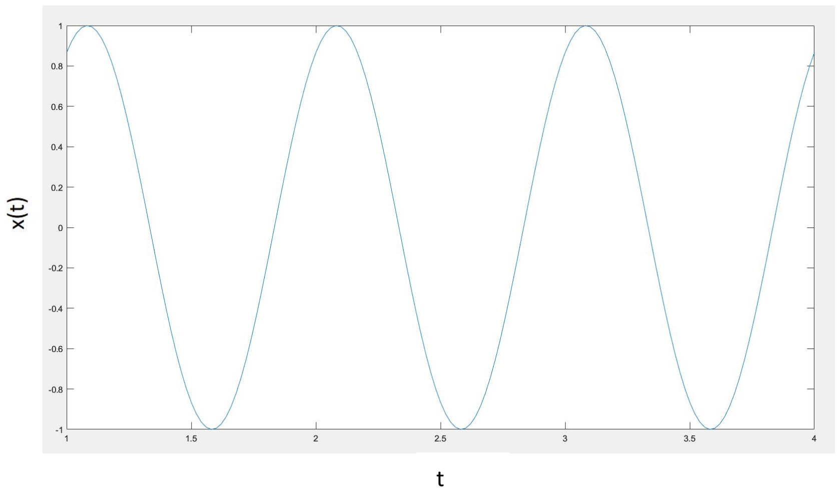


Fig.1 Plot of part q

S) Using the same formula: [(1-0)/0.01]+1, we get 101 time points and using the linspace(0,1,101) command, the new signal can be added to the previous plot.

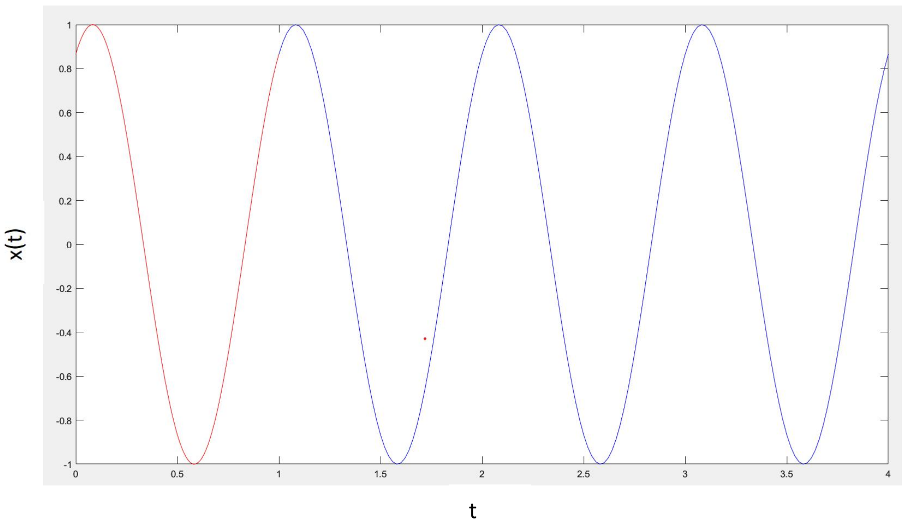


Fig.2 Plot of part s

t)

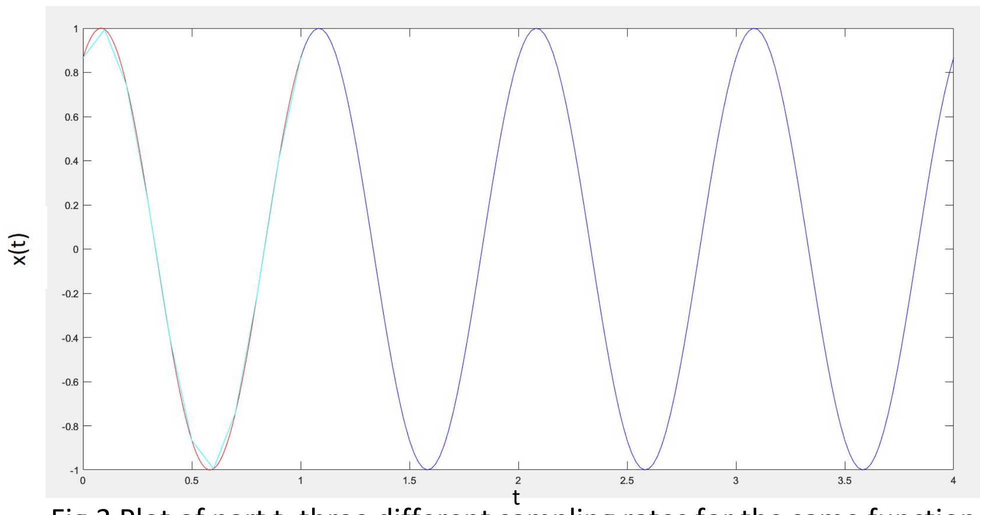


Fig.3 Plot of part t, three different sampling rates for the same function

u)

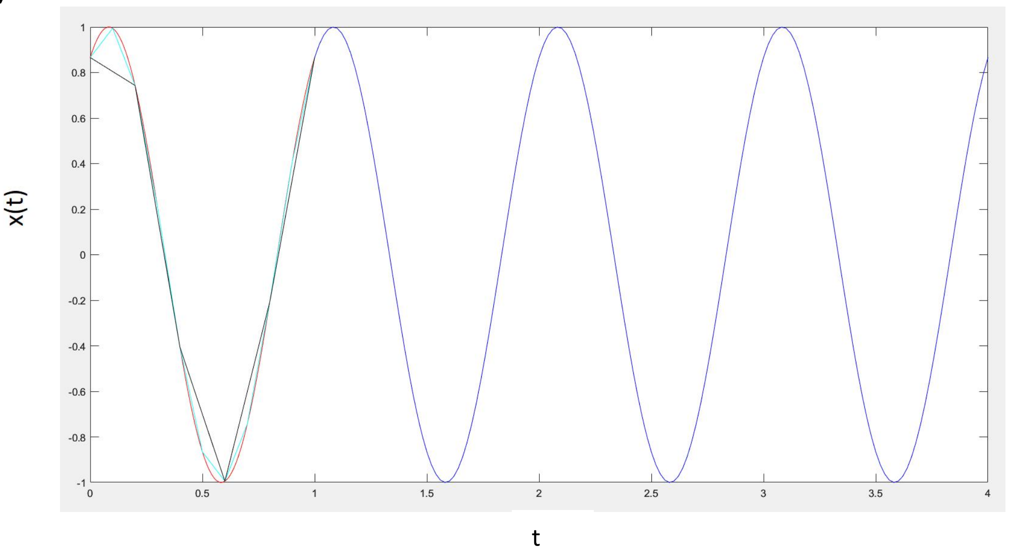


Fig.4 Plot of part u, four different sampling rates for the same function

V) Looking at the plots, it can be seen that the red line with 0.01 step size is more likely to represent the continuous x(t). As the time difference between two consecutive time points goes to zero, the likelihood to represent a continuous signal increases. Since with zero time difference, an infinite set of time points will be used for sampling which will ideally connect every x value corresponding to an infinite number of t’s.

W) The plot command uses lines for the connection of data points.

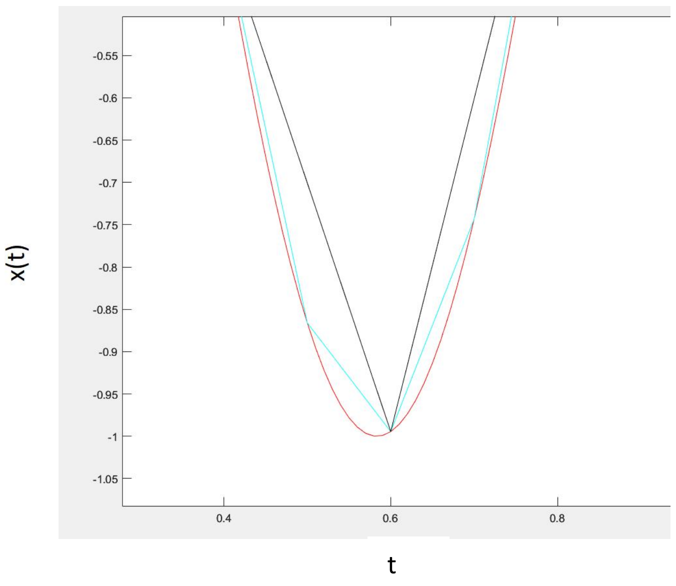


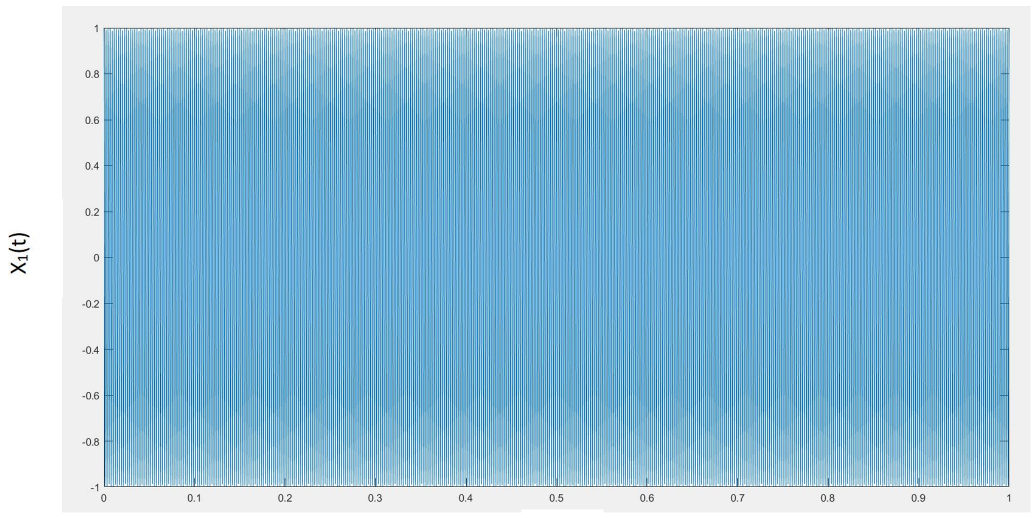
Fig.5 Connection of data points with lines.

x) With stem command discrete values of the x(t) are displayed, unlike the plot command the data points are not connected with lines

Part 2

a) Sound command plays vectors as sounds, it sends the signal to the speaker as sound with the default sample rate of 8192 Hz unless a special frequency is specified. It assumes that the values of the signal is between [-1,1] and values outside this range are clipped automatically. Soundsc command functions similarly however instead of clipping the out-of-range values, it scales the values so that the data can be played as loud as possible. For the Cosine signal both commands can be used since the values of cosine are always between [-1,1] which is the appropriate range for both commands.

b) The sound produced is the same for sound and soundsc commands.



t

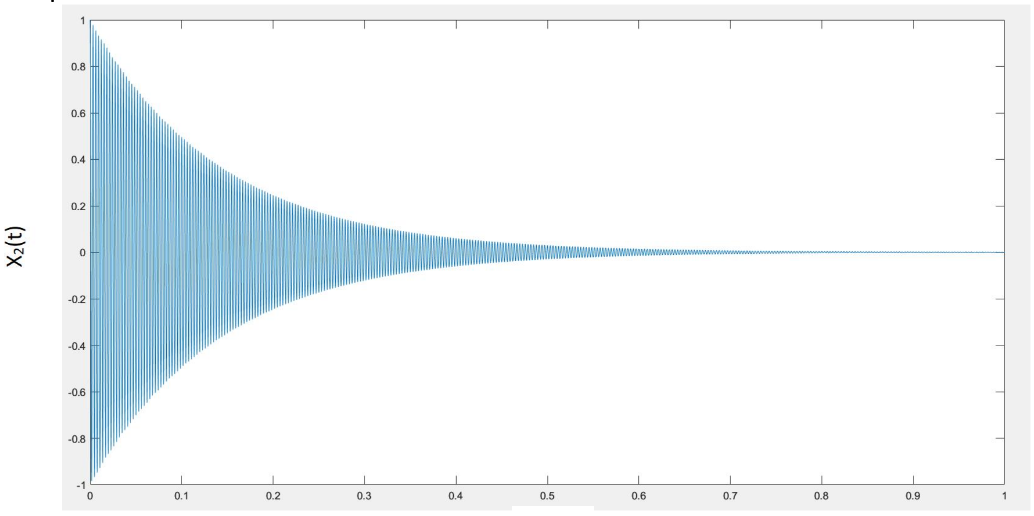
Fig.6 Signal x1 with frequency 440 Hz

c) The pitch of the sound increases as the frequency increases, high frequencies correspond to high pitched sounds.

The code for the signal x2(t):

```
t=[0:1/8192:1];
f=330;
a=7;
x2 = exp(-a*t).*cos(2*pi*f*t);
plot(t,x2)
sound(x2)
```

The plot of x2 versus t:



t

Fig.7 Signal x2 with frequency 300 Hz and a=7

The sound of the signal x1 is “steady” , in other words the maximum magnitude of the signal amplitude is 1 and doesn’t change over time. However when this signal is multiplied with a decaying exponential, the cosine function becomes multiplied with a decaying envelope hence the maximum amplitude of the signal decays over time and the sound fades out over time. The sound of the signal x1 resembles the sound produced by a flute since blowing constant air creates steady sound and the sound of the signal x2 is like a piano’s sound since after pressing a piano key the sound fades out over time.

For a=2, the plot of x2 versus t:

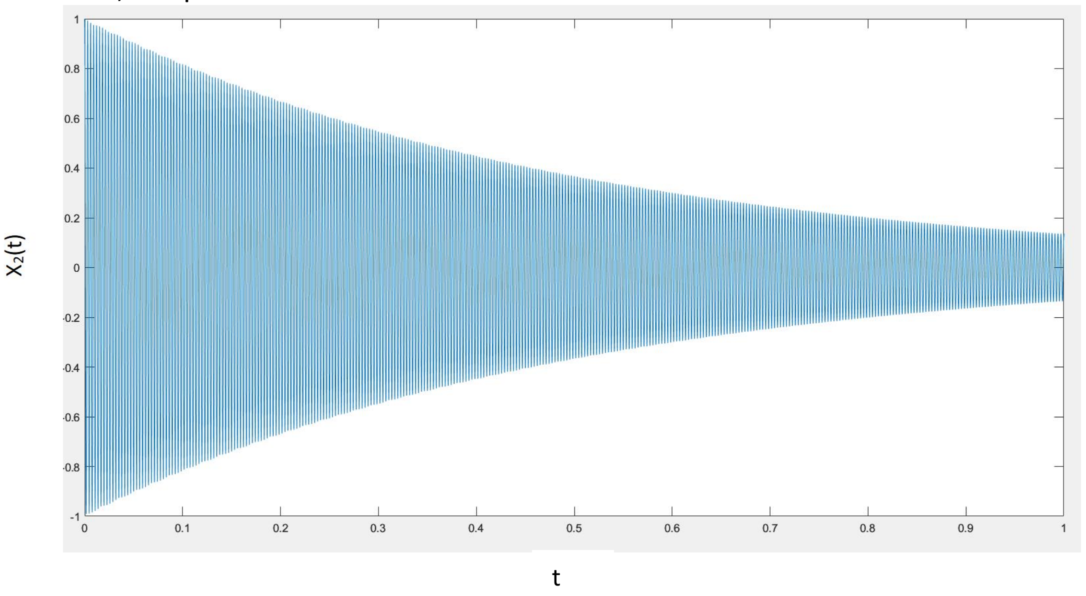


Fig.8 Signal x2 with frequency 300 Hz and a=2

For a=11, the plot of x2 versus t:

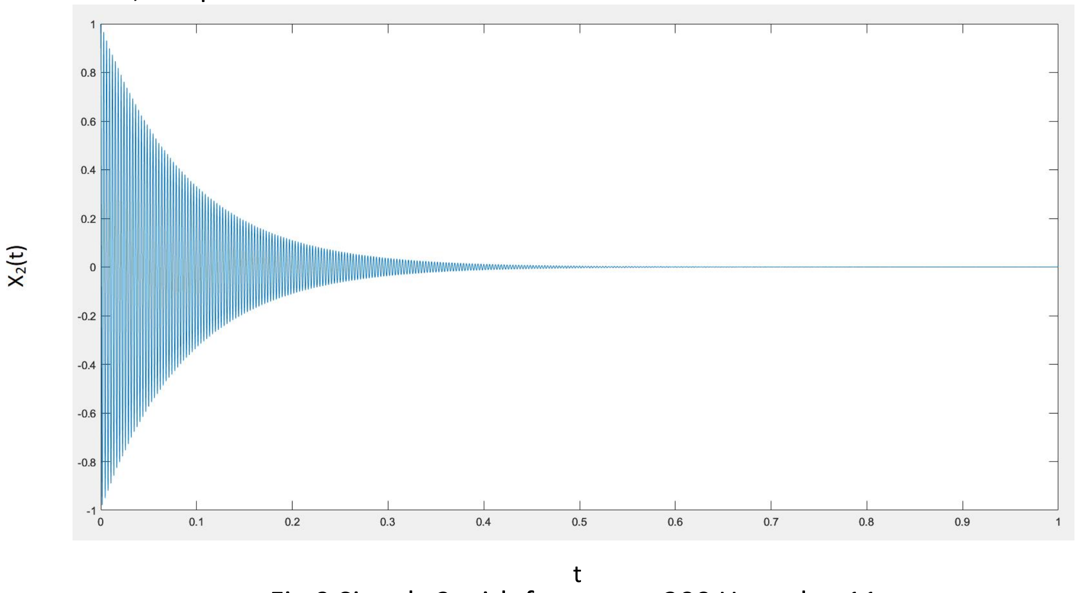


Fig.9 Signal x2 with frequency 300 Hz and a=11

Comparing Figures 7,8 and 9 it can be said that as the value “a” increases, the sound fades out more quickly and the hearing duration becomes shorter. The number a determines the decaying rate of the exponential, multiplying a faster decaying exponential with a sinusoidal signal causes shorter life time for that sinusoidal signal.

The code for the signal x3(t):

```
t=[0:1/8192:1];
f0= 510;
f1= 4;
x3= cos(2*pi*f1*t).*cos(2*pi*f0*t);
plot(t,x3)
sound(x3)
```

The plot of x3 versus t when f0=510Hz and f1=4Hz:

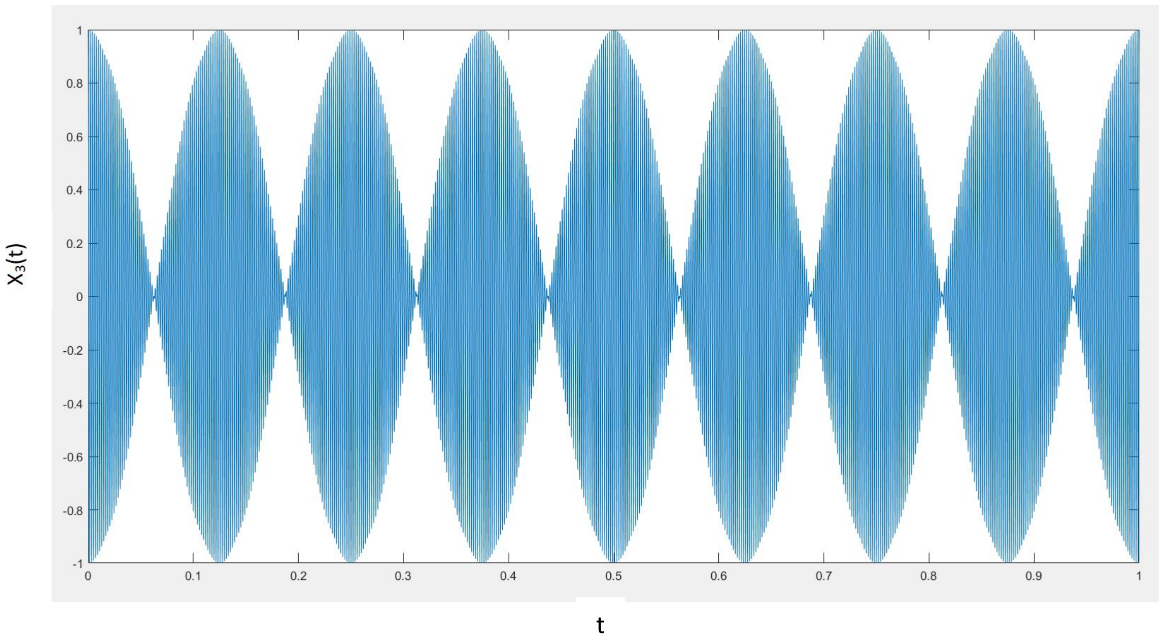


Fig.10 Signal x3 with f0=510 Hz and f1=4Hz

The low frequency cosine acts as a amplitude envelope for the higher frequency cosine signal. The signal x1 was fading out over time since it was multiplied with a decaying exponential, now the signal x3 does not fade out since there isn’t any decaying scaling factor however this time the lower frequency cosine function scales the amplitude of the higher frequency cosine function. The sound appears to be rising and falling continuously due to the lower frequency cosine function.

For f1=2Hz, the plot of x3 versus t:

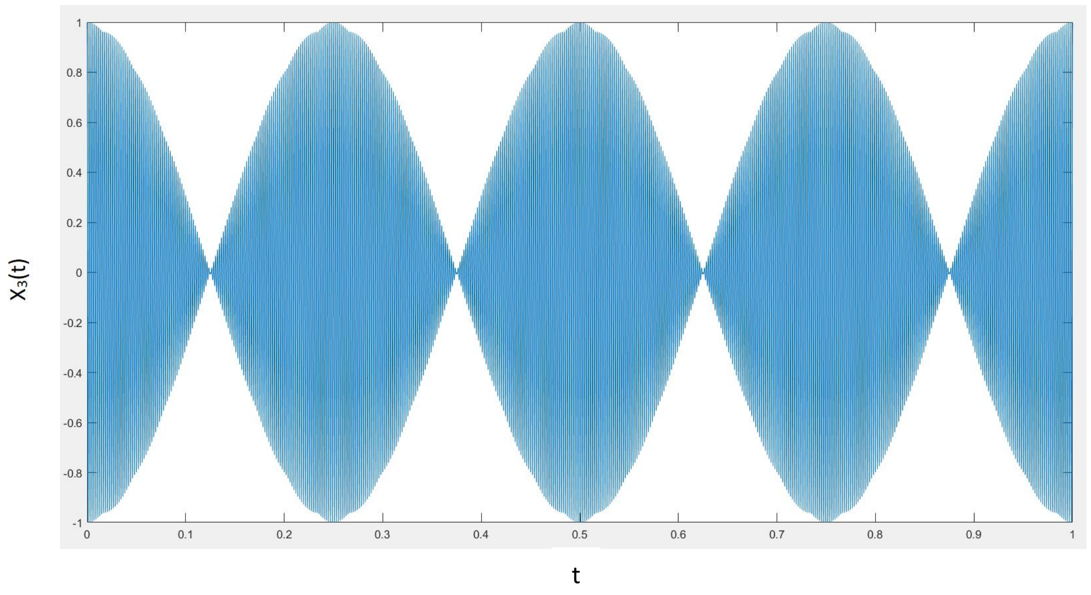


Fig.11 Signal x3 with f0=510 Hz and f1=2Hz

For f1=6Hz, the plot of x3 versus t:

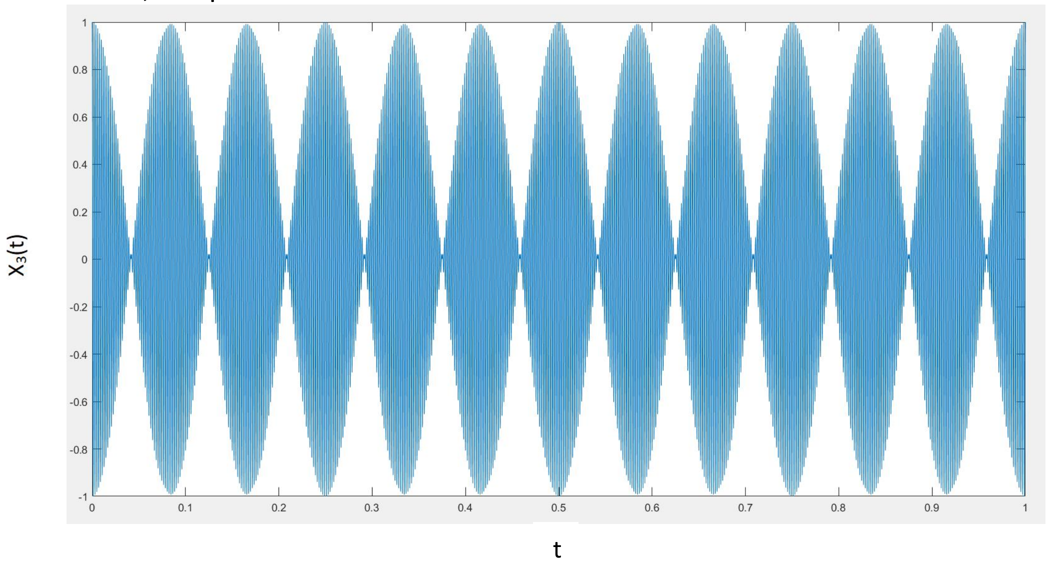


Fig.12 Signal x3 with f0=510 Hz and f1=6Hz

When the frequency of the lower cosine function is increased the frequency of the amplitude envelope increases and the sound starts fluctuate faster.

Using the trigonometric identity (2) the signal x3 (1) can be written as the addition of two cosines

$$x_3(t) = \cos(2\pi f_1 t)\cos(2\pi f_0 t) \qquad (1)$$

$$\cos(x) + \cos(y) = 2\cos(\frac{x+y}{2})\cos(\frac{x-y}{2}) \qquad (2)$$

The right hand side of (2) resembles (1) hence we get:

$$\frac{x+y}{2} = 2\pi f_1 t \quad (3)$$

$$\frac{x-y}{2} = 2\pi f_0 t \quad (4)$$

From here we find x as $2\pi t(f_1 + f_0)$ and y as $2\pi t(f_1 - f_0)$ hence the signal (1) can be written as (5)

$$x_3(t) = \frac{1}{2}[\cos(2\pi t(f_1 + f_0)) + cos(2\pi t(f_1 - f_0))] \quad (5)$$

Part 3

The signal $x_1(t) = cos(2\pi f_0 t)$ can be written of the form $x(t) = cos(2\pi\emptyset(t))$ where $\emptyset(t)$ is equal to $f_0 t$. Hence the time derivative of $\emptyset(t)$ will give the instantaneous frequency.

$$f_{ins} = \frac{d\phi(t)}{dt} = \frac{df_0 t}{dt} = f_0 \quad (6)$$

For the signal $x_4(t) = cos(\pi\alpha t^2)$ the $\emptyset(t)$ corresponds to αt^2. Similarly by using the relations in (6) the instantaneous frequency can be found as:

$$f_{ins} = \frac{d\phi(t)}{dt} = \frac{d\alpha t^2}{dt} = \alpha t \quad (7)$$

At t=0, f_{ins} =0, and at t= t_o the corresponding frequency is αt_0.
Matlab code for the signal $x_4(t) = cos(\pi\alpha t^2)$:

```
t=[0:1/8192:1];
a= 1777;
x4= cos(pi*a*t.^2);
plot(t,x4)
sound(x4)
```

The corresponding plot:

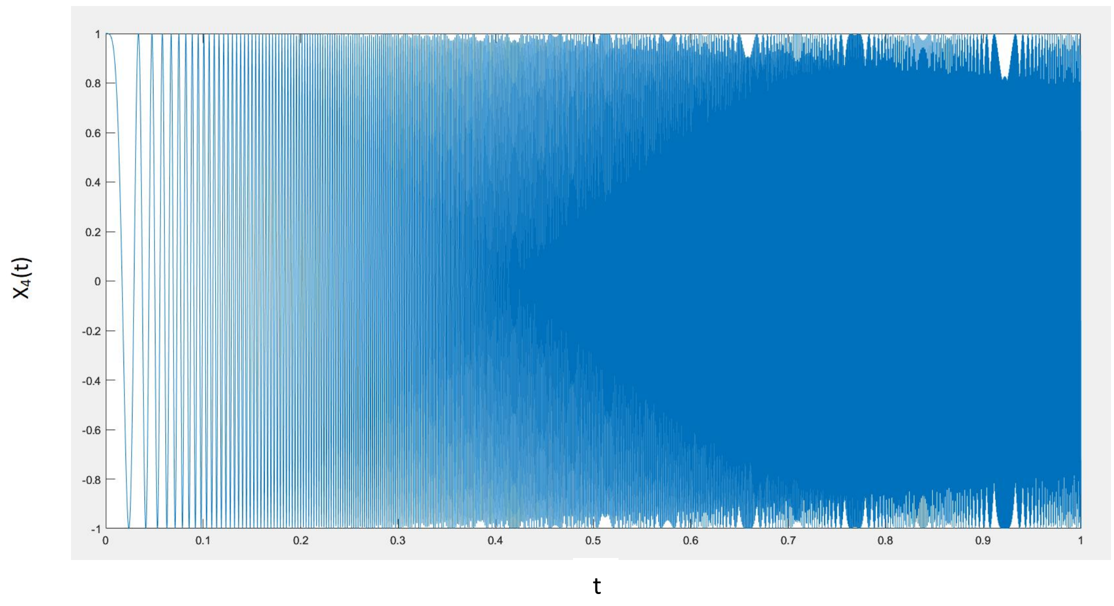


Fig.13 Signal x4 with a=1777

With the chosen “α” value, the corresponding instantaneous frequencies will depend on the product α*t as explained in (7). Since t changes from 0 to 1, αt values will be between [0,1777]. As the frequency increases linearly over time, the sound becomes higher pitched. Different values of α, corresponds to different frequency ranges. While keeping the time interval constant; if we change α to a higher value, the rate of change of the frequency values will increase since we are trying to fit a bigger frequency interval into the same time interval. In physical world this situation will cause the sound to get higher pitched faster. Contrarily if we use lower α values, the frequency rate of change will be slower e.g. the slope of the linear relation between time and instantaneous frequency will decrease. Examples of these situations can be seen below.

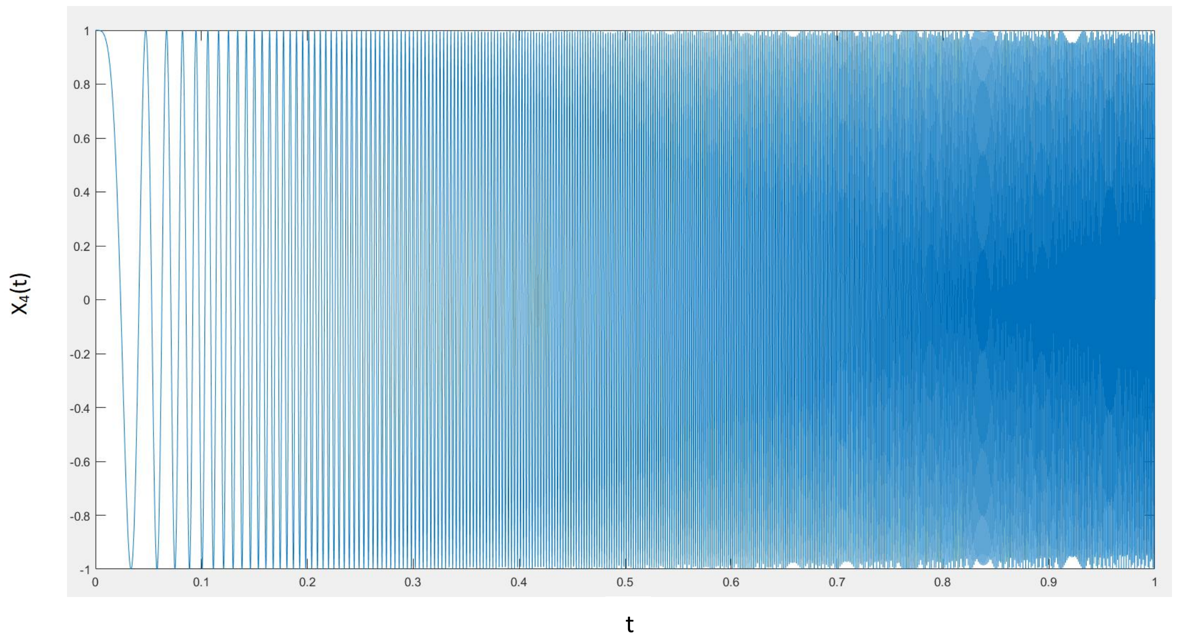


Fig.14 Signal x4 with a=1777/2

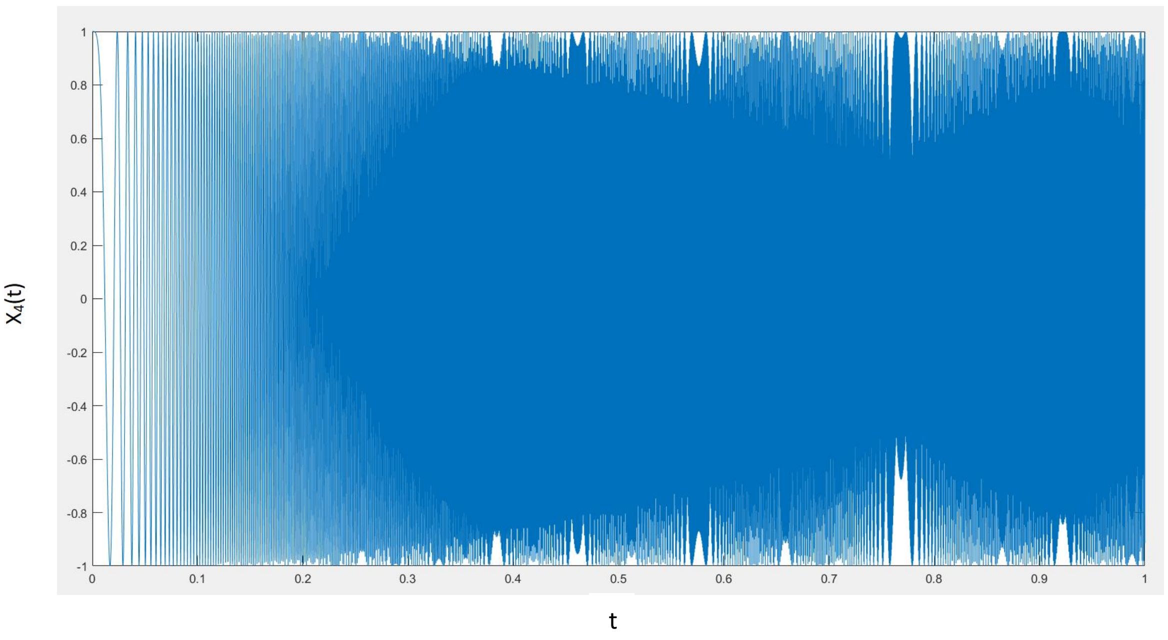


Fig.15 Signal x4 with a=1777*2

Matlab code for $x_5 = \cos(2\pi(-500t^2 + 1600t))$:

```
t=[0:1/8192:2];
x5= cos(2*pi*(-500*t.^2+1600*t));
plot(t,x5)
sound(x5)
```

The instantaneous frequency for x5 can be found using (6) as well. For this case the $\emptyset(t)$ corresponds to $-500t^2 + 1600t$. The instantaneous frequency can be found as

$$f_{ins} = \frac{d\phi(t)}{dt} = \frac{d(-500t^2 + 1600t)}{dt} = -1000t + 1600 \qquad (8)$$

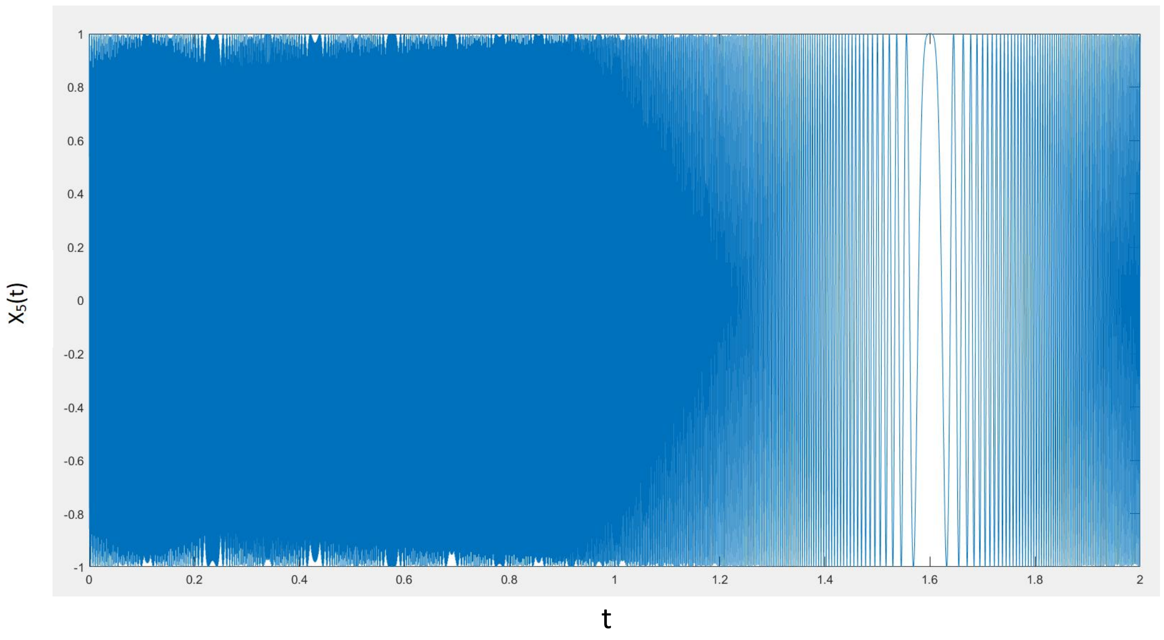


Fig.16 Signal x5

Using (8) we can find the following frequency values:
- at t=0, f=1600 Hz
-at t=1, f=600 Hz
-at t=2, f= -400 Hz

As t increases, the frequency decreases since the slope of their relation is negative, however from Fig.16 and from the sound, it can be both seen a and heard that after some time, the absolute value of the frequency starts to increase again. This is due to the fact that when t=1.6, f_{ins} becomes zero and after 1.6 seconds it starts to grow in negative direction. Hence the sound starts to get high pitched again.

Part 4

The code for $x = \cos(2\pi\alpha t + \varphi)$:

```
t=[0:1/8192:1];
a= 1777;
phi=pi/4;
x=cos(2*pi*a*t+phi);
sound(x)
plot(t,x)
```

The change in the phi value does not effect the pitch of the sound, for all trials the sound is the same, it only changes the phase. f_{ins} does not depend on the phi value. This can be explained again by using property (6) but first some algebraic operations should be done:

$$\cos(2\pi\alpha t + \varphi) = \cos(2\pi(\alpha t + \frac{\varphi}{2\pi})) \qquad (9)$$

$$\phi(t) = \alpha t + \frac{\varphi}{2\pi} \qquad (10)$$

$$f_{ins} = \frac{d\phi(t)}{dt} = \frac{d(\alpha t + \frac{\varphi}{2\pi})}{dt} = \alpha \qquad (11)$$

As can be seen from Eq. 11, the instantaneous frequency only depends on the variable α.

Part 5

$x_1(t) = A_1\cos(2\pi f_0 t + \phi_1)$ $\quad A_1 \geqslant A_2 \geqslant 0$

$x_2(t) = A_2\cos(2\pi f_0 t + \phi_2)$

$x_3(t) = x_1(t) + x_2(t)$

$\hookrightarrow A_3\cos(2\pi f_3 t + \phi_3)$, $A_3 \geqslant 0$

$x_1 + x_2 = A_1\cos(2\pi f_0 t + \phi_1) + A_2\cos(2\pi f_0 t + \phi_2) = A_3\cos(2\pi f_3 t + \phi_3)$

$\alpha e^{j\phi} = \alpha\cos\phi + \alpha j \sin\phi$, $Re\{\alpha e^{j\phi}\} = \alpha\cos\phi$ $\Rightarrow$ using these relations:

$Re\{A_1 e^{j(2\pi f_0 t + \phi_1)}\} = A_1\cos(2\pi f_0 t + \phi_1)$

$Re\{A_2 e^{j(2\pi f_0 t + \phi_2)}\} = A_2\cos(2\pi f_0 t + \phi_2)$

$Re\{A_3 e^{j(2\pi f_0 t + \phi_3}\} = A_3\cos(2\pi f_3 t + \phi_3)$

$$\boxed{Re\{A_1 e^{j(2\pi f_0 t + \phi_1)}\} + Re\{A_2 e^{j(2\pi f_0 t + \phi_2)}\} = Re\{A_3 e^{j(2\pi f_3 t + \phi_3)}\}}$$

$A_1 e^{j(2\pi f_0 t + \phi_1)} + A_2 e^{j(2\pi f_0 t + \phi_2)} = e^{j(2\pi f_0 t)}(A_1 e^{j\phi_1} + A_2 e^{j\phi_2})$

$e^{j(2\pi f_0 t)}(A_1 e^{j\phi_1} + A_2^{j\phi_2}) = A_3 e^{j(2\pi f_3 t)} e^{j\phi_3}$

they should be equal, Hence $\boxed{f_0 = f_3}$ (1)

Other elements should also be equal : $A_1 e^{j\phi_1} + A_2 e^{j\phi_2} = A_3 e j^{\phi_3}$

$A_1\cos\phi_1 + A_1 j \sin\phi_1 + A_2\cos\phi_2 + A_2 j \sin\phi_2 = A_3\cos\phi_3 + A_3 j \sin\phi_3$

Both imaginary and real parts should be equal :

$$\begin{cases} A_1\cos\phi_1 + A_2\cos\phi_2 = A_3\cos\phi_3 \\ A_1\sin\phi_1 + A_2\sin\phi_2 = A_3\sin\phi_3 \end{cases} \Rightarrow \cot\phi_3 = \frac{A_1\cos\phi_1 + A_2\cos\phi_2}{A_1\sin\phi_1 + A_2\sin\phi_2}$$

(2) $\boxed{\phi_3 = \cot^{-1}\left(\frac{A_1\cos\phi_1 + A_2\cos\phi_2}{A_1\sin\phi_1 + A_2\sin\phi_2}\right)}$

Squaring both sides:

$$(A_1\cos\phi_1 + A_2\cos\phi_2)^2 = A_3^2\cos\phi_3^2$$
$$+ \quad (A_2\sin\phi_1 + A_2\sin\phi_2)^2 = A_3^3\sin\phi_3^2$$

$$A_3^2\underbrace{(\sin\phi_3^2 + \cos\phi_3^2)}_{1} = (A_1\cos\phi_1 + A_2\cos\phi_2)^2 + (A_1\sin\phi_1 + A_2\sin\phi_2)^2$$

$$A_3 = \sqrt{\underbrace{(A_1\cos\phi_1 + A_2\cos\phi_2)^2}_{} + \underbrace{(A_1\sin\phi_1 + A_2\sin\phi_2)^2}_{}}$$

$$A_1^2\cos\phi_1^2 + 2A_1A_2\cos\phi_1\cos\phi_2 + A_2^2\cos\phi_2^2 + A_1^2\sin\phi_1^2 + 2A_1A_2\sin\phi\sin\phi_2 + A_2^2\sin\phi_2^2$$

$$A_1^2\underbrace{(\cos\phi_1^2 + \sin\phi_1^2)}_{1} + A_2^2\underbrace{(\cos\phi_2^2 + \sin\phi_2^2)}_{1} + 2A_1A_2(\underbrace{\cos\phi_1\cos\phi_2 + \sin\phi_1\sin\phi_2}_{\cos(\phi_1-\phi_2)})$$

$$\boxed{A_3 = \sqrt{A_1^2 + A_2^2 + 2A_1A_2\cos(\phi_1-\phi_2)}} \quad (3)$$

a) A_1 and A_2 are positive numbers therefore the $\cos(\phi_1-\phi_2)$ term should be minimum for A_3 to be minimum. $\min\cos(x) = 0$

$$\min\cos(\phi_1-\phi_2) \Rightarrow \phi_1-\phi_2 = \frac{\pi}{2} + \pi k \quad , k \in \mathbb{Z}$$

b) For maximum value of A_3 cosine term should be 1.

$$\max\cos(\phi_1-\phi_2) \Rightarrow \phi_1-\phi_2 = 2\pi k \quad , k \in \mathbb{Z}$$

$$\max A_3 = \sqrt{A_1^2+A_2^2+2A_1A_2}$$
$$\min A_3 = \sqrt{A_1^2+A_2^2}$$