EEE - 321: Signals and Systems

Lab Assignment 1

Yiğit Narter

22102718

Section 02

Part 1

a) When a = [3.2 34/7 -6 24] is typed, the variable is created as a row vector. When a = [3.2; 34/7; -6; 24] is typed, this time the variable a is a column vector. The variable is displayed on the console in both cases.

b) With this code, the variable "a" is assigned a row vector, and a variable "b" is created as a column vector. The difference from part a is that the variables aren't displayed on the console due to the ";" placed at the end.

c) When a semicolon is placed, the result of the command is not displayed on the console. This helps shorten the elapsed time. If the code deals with large data, it is useful to use semicolons, since it would take more time to display the large data on the console.
By using the tic and toc commands, the elapsed time for the "a = [3.2 34/7 -6 24]" command is measured as 0.012687 seconds, while it is 0.000181 seconds for the "a = [3.2 34/7 -6 24];" command.

d) "Error using *" message is displayed. The error occurs because the dimensions of the matrices don't match. For matrix multiplication, the number of columns in the first matrix should match the number of rows in the second matrix. To operate on each element of the matrix individually, ".*" should be used (for elementwise multiplication).

e) The resulting vector c is displayed as:

```
c =
   1.0e+03 *

    0.0186    0.0233   -0.0300   -2.4480
```

This time, by adding a dot in front of *, the operation is elementwise multiplication. It means that the first element of a is multiplied by the first element of b, the second with the second, and so on. Since elementwise multiplication is commutative, changing the orders of a and b does not change the result.

f) This time since a is a row vector (1x4) and b is a column vector (4x1), matrix multiplication is applicable (the number of columns in a matches the number of rows in b). The result should be a 1x1 matrix since a's row number and b's column number are both 1. The result is displayed as:

```
c =

  -2.4361e+03
```

g) Again, matrix multiplication is performed. This time, a is a column vector (4x1) and b is a row vector (4x1). Therefore, the result should be a 4x4 matrix. The result is displayed as:

```
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
```

h) A row vector is created. The first element is 1, and the elements are increased by 0.01 step size. The last element is 2, and in total there are 101 elements. The vector is a = [1 1.01 1.02 … 1.98 1.99 2] and it is displayed on the console.

i) The elapsed time is 0.000394 seconds.

j) The elapsed time is 0.003631 seconds.

k) The elapsed time is 0.000787 seconds. It can be seen that the elapsed times can be compared as j>k>i. Therefore, the built-in function in part i is the most ideal and efficient one due to the shortest elapsed time.

l) b is a vector containing the sine values of each element in vector a. MATLAB computes each element's sine value in a, which contains angles starting from 0 up to 2*pi with a step size of pi/8, and puts the values in vector b.

m) In plot(x), the values of x are plotted in the y-axis, and the x-axis shows the index of the corresponding element in vector x. For example, the 151 on the x-axis actually corresponds to the last element in x (which contains 151 elements in total), and the y-axis shows its corresponding value for the function. The interval on the x-axis starts from 1 to the number of rows in x, in other words, length(x) which is 151.
In plot (t,x), the x-axis shows the element values in t this time, and again the y-axis displays the values in vector x. In plot(x,t), the order is reversed, the y-axis shows the elements in t and the x-axis shows the elements in vector x.

n) When plot(t,x,'-+') is typed in, the plot marks the points on the graphs with a "+" sign. These points are the elements in vector t. The previous graph, which essentially connects all these points and displays the function as continuous, is also present. However, when plot(t,x,'+') is typed in, the continuous graph disappears and only the "+" signs are present.

o) There are 26 points included in t. It is a 1x26 row matrix.

p) By typing `t = linspace(0, 1, 26)` in which the start point, the end point, and the number of elements are specified, the variable can be generated.

q) The resulting vector x, just like t, is a 1x26 row matrix. x = [0.8660 0.9632 0.9998 … 0.5180 0.7145 0.8660].

r) The resulting graph is shown below.

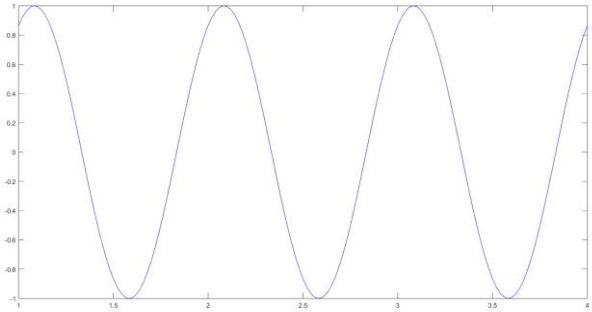


Figure 1: Graph for part r

s) This time, there are 101 elements in t. The resulting graph is shown below.

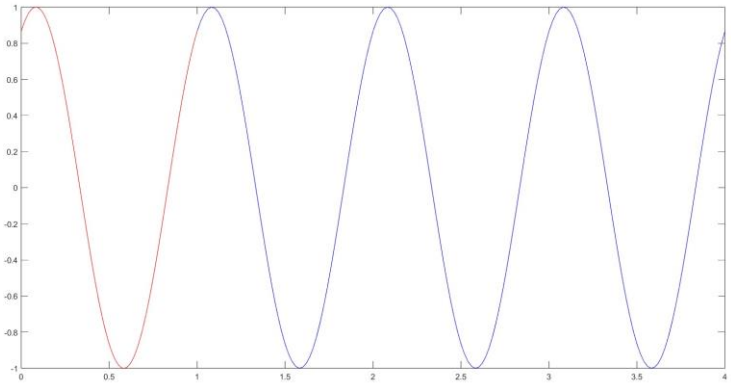


Figure 3: Graph for part s, now a red curve is added to the plot

t) The resulting graph is shown below.

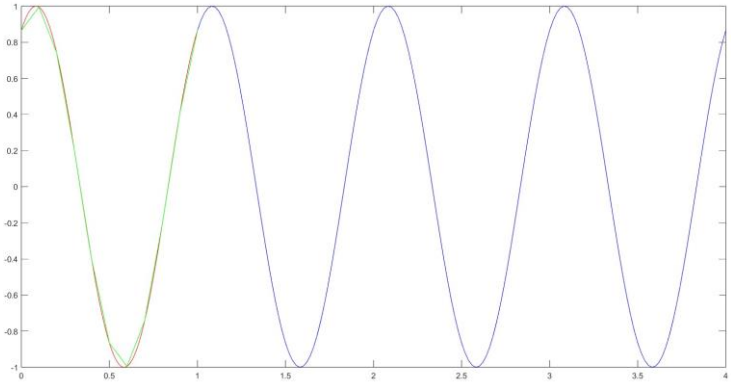


Figure 4: Graph for part t, now a green curve is added

u) The resulting graph is shown below.

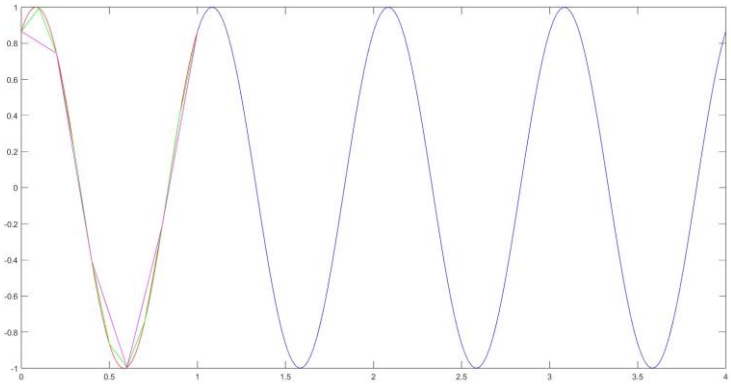


Figure 5: Graph for part t, now a magenta curve is added

v) The red curve, which is in part s, is the most likely for the continuous function x(t). The reason is that as the interval (step size) decreases, the accuracy increases and the plot approaches to a curve-like shape. The smallest step size is in the vector of part s, which is 0.01.
w) The points are connected by the "plot" command. This makes the plot resemble a continuous function. However, with bigger step sizes, the connection accuracy decreases.
x) The "stem" command plots the function as a discrete signal, meaning that the values of the points are displayed separately without being connected as in the "plot" command. The "plot" command plots as a continuous function.

Part 2

a) The "sound" command sends the audio signal to the speaker with 8192 Hz sample rate, which is the default value. The "soundsc" command scales the values of the audio signal such that they fit in the range of -1.0 to 1.0 before sending it to the speaker with the default sample rate of 8192 Hz. This helps the signal being played as loudly as possible.
Both are appropriate to listen to the discrete version. The signal is only heard louder with the "soundsc" command.

b) f_0=440. The code used in this part:

```
t=[0:1/8192:1];
f0=440;
x1 = cos(2*pi*f0*t);

plot(t,x2)
sound(x2)
```

The plot is shown below.

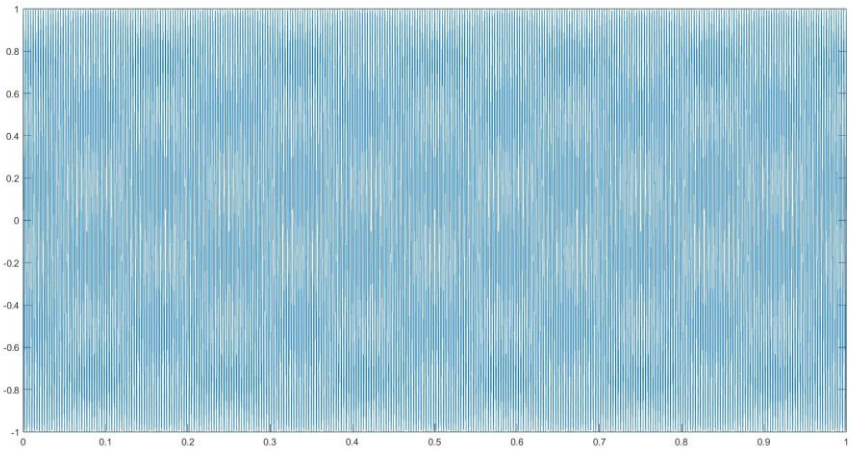


Figure 6: x1(t) versus t

c) f_0=687
d) f_0=883

As the frequency increases, the pitch of the sound also increases.

- To obtain $x_2(t)$:

```
x2 = exp(-a*t).*cos(2*pi*f0*t);
```

By adding the e^{-at}, the signal is now a decaying sinusoid. The signal's amplitude decays as time passes. Because of this, the sound of x_2 vanishes and the volume decreases (which is related to the amplitude), while x_1's sound is heard as constant. The resulting graph is shown below.

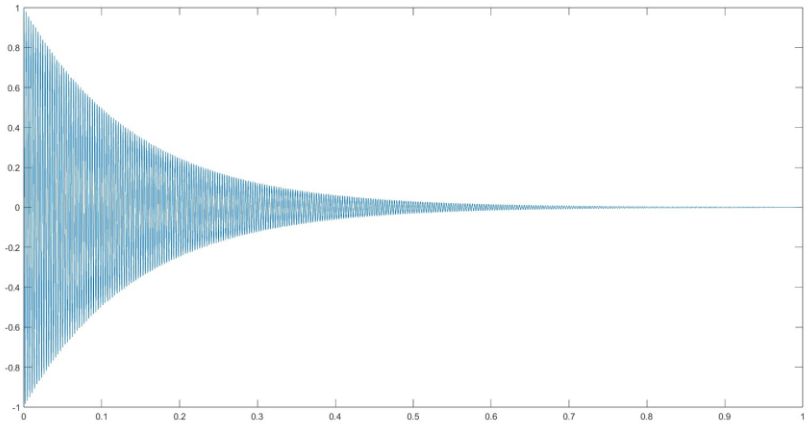


Figure 7: x2(t) versus t

The sound of x_2 resembles the piano more since the sound in pianos is produced by the strings and these sounds are also exponentially decaying. For x_1, the sound resembles a flute, in which the sound remains constant as long as the player blows into it. As the player blows, there is no decaying just like in the x_1 signal.

For x_2, as a is increased, the duration of the sound decreases. This is because the time constant (τ) decreases with increasing a, which makes the signal require less time to approach zero (It takes approximately 5 time constants to get to zero). The time constant indicates the rate at which the signal is decaying. As it increases, the signal takes more time to decay. The time constant τ can be found as:

$$x_2(t) = e^{-at}\cos(2\pi f_0 t) = e^{-\frac{t}{\tau}}\cos(2\pi f_0 t) \;\Rightarrow\; \tau = \frac{1}{a}$$

- To obtain $x_3(t)$:

```
x3 = cos(2*pi*f0*t).*cos(2*pi*f1*t);
```

The plot is shown below:

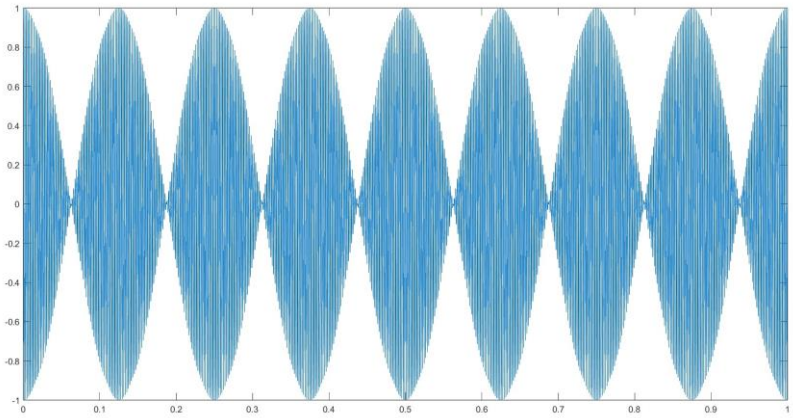


Figure 8: x3(t) versus t

The difference between x_1 and x_3 is that x_3's sound changes periodically over time. Its frequency oscillates. This periodical change of the frequency is the effect of the low-frequency (f_1) cosine term in x_3. As f_1 is increased, the frequency is changed more frequently and the changes in the sound are also more frequent. To write x*3* as a sum of two cosines we can use the identity:

$$\cos(a)\cos(b) = \frac{\cos(a+b) + \cos(a-b)}{2}$$

Putting $a=2\pi f_1 t$ and $b=2\pi f_0 t$, we obtain:

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

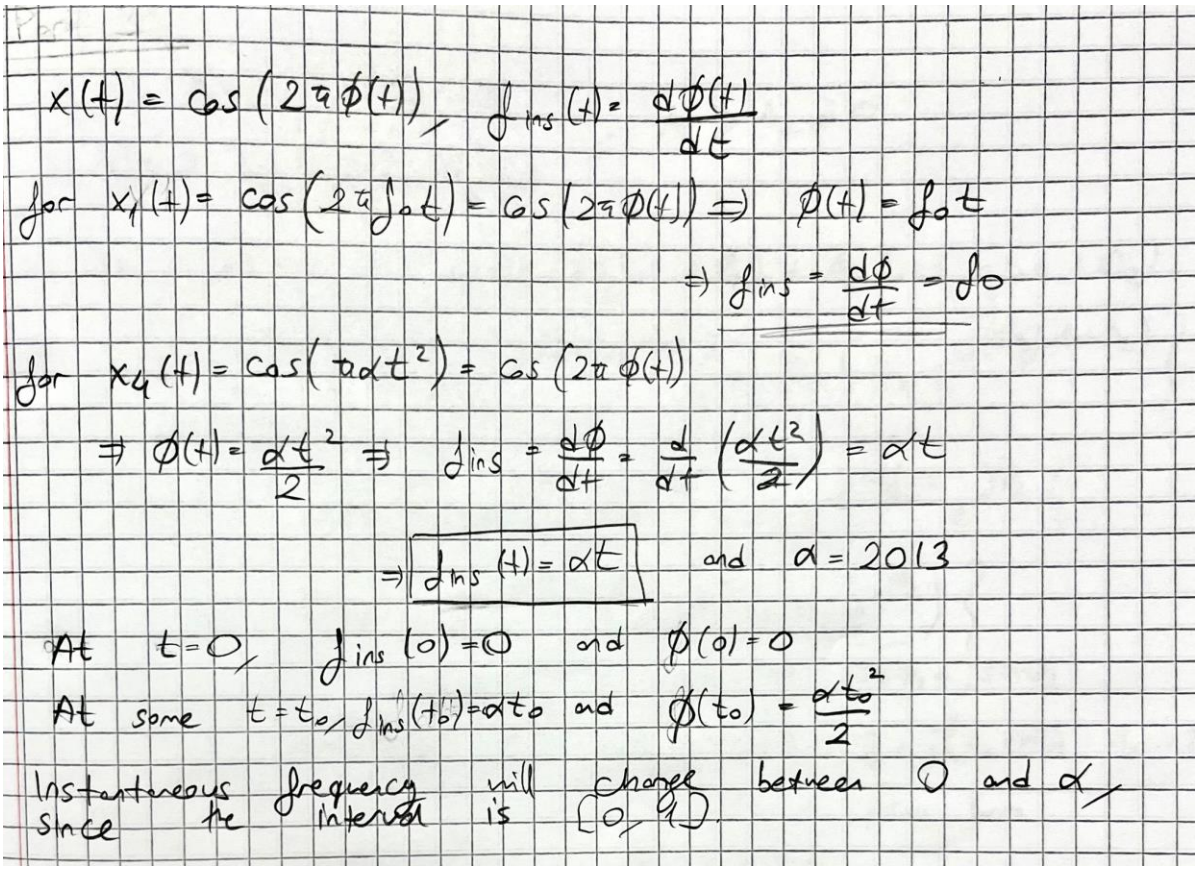
Since $f_1 << f_0$ for our case, this is a summation of two cosines with very similar (nearly equal) frequencies. This creates a phenomenon known as "beat". The sound is therefore heard with a changing pitch.

Part 3

- To obtain $x_4(t)$:

```
t=[0:1/8192:1];
alpha = 2013;
x4 = cos(alpha * pi * t.^2);
```

From the website www.random.org, α is generated as 2013.

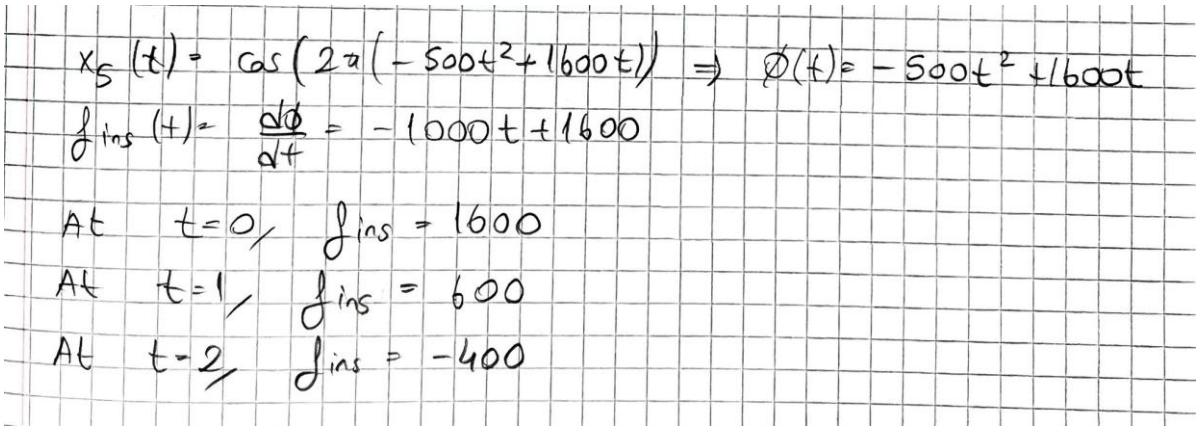


Because the frequency is increasing linearly, the sound's pitch also increases. Instantaneous frequency is a commonly used concept in signal processing. They are used to represent and analyze time-varying functions. Instantaneous frequency is used in lasers, like single-frequency lasers. It is also important in music. In music scores, notes are essentially a specified instantaneous frequency value for a specific time interval. This concept is also used in Doppler effect application which also deals with changes in frequencies of different waves.

As α is increased, the signal gets dense more quickly. Hence, the sound gets to a higher pitch at a shorter time.

- To obtain $x_5(t)$:

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



As time goes on, the frequency is decreasing. However, after the point t=1.6, the instantaneous frequencies start taking negative values. Since cosine is an even function, these negative values actually give the same sound as their positive ones. Therefore, the frequency of the sound decreases, then it starts increasing for a short time.

Part 4

In this part, by changing the phi, the function is shifted with that phase. Since cosine is a periodic function, the volume and the pitch heard stays the same as phi is changed. The volume doesn't change since the amplitude stays the same. The pitch doesn't change since the frequency stays the same.

The code used in this part:

```
a = 2013;
t=[0:1/8192:1];
phi = pi;

x = cos(2*pi*a*t + phi);

sound(x)
```

Part 5

$x_1(t) = A_1 \cos(2\pi f_0 t + \phi_1)$

$x_2(t) = A_2 \cos(2\pi f_0 t + \phi_2)$, $A_1 \geq A_2 \geq 0$

$x_3(t) = x_1(t) + x_2(t) = A_1 \cos(2\pi f_0 t + \phi_1) + A_2 \cos(2\pi f_0 t + \phi_2)$

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

$= \text{Re}\{A_1 e^{j2\pi f_0 t} \cdot e^{j\phi_1} + A_2 e^{j2\pi f_0 t} e^{j\phi_2}\}$

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

$A_1 e^{j\phi_1} = \cos\phi_1 + j\sin\phi_1$, $A_2 e^{j\phi_2} = \cos\phi_2 + j\sin\phi_2$

$= \text{Re}\{e^{j \cdot 2\pi f_0 t} \cdot (\underbrace{A_1\cos\phi_1 + A_2\cos\phi_2 + j(A_1\sin\phi_1 + A_2\sin\phi_2)}_{\text{in polar form}})\}$

has magnitude:

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

and argument

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

$= \text{Re}\{e^{j2\pi f_0 t} \cdot A_3 \cdot e^{j\phi_3}\}$

$= \text{Re}\{A_3 \cdot e^{j(2\pi f_0 t + \phi_3)}\} = A_3\cos(2\pi f_0 t + \phi_3) = \underbrace{A_3\cos(2\pi f_3 t + \phi_3)}_{\text{given in question}}$

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

$f_3 = f_0$

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

To further simplify A_3:

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

$$= \Big(A_1^2\cos^2\phi_1 + 2A_1A_2\cos\phi_1\cos\phi_2 + A_2^2\cos^2\phi_2 + A_1^2\sin^2\phi_1 + 2A_1A_2\sin\phi_1\sin\phi_2 + A_2^2\sin^2\phi_2\Big)^{1/2}$$

$$= \Big(A_1^2\underbrace{(\cos^2\phi_1 + \sin^2\phi_1)}_{1} + A_2^2\underbrace{(\cos^2\phi_2 + \sin^2\phi_2)}_{1} + 2A_1A_2\underbrace{(\cos\phi_1\cos\phi_2 + \sin\phi_1\sin\phi_2)}_{\text{sum formula } \cos(\phi_1-\phi_2)}\Big)^{1/2}$$

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

a) A_3 is minimum when the $\cos(\phi_1-\phi_2)$ takes its minimum value, which is -1

to have $\boxed{\cos(\phi_1-\phi_2) = -1}$, we need $\boxed{\phi_1-\phi_2 = (2n+1)\pi,\ n\in\mathbb{Z}}$

and $A_{3,min} = \sqrt{A_1^2 + A_2^2 - 2A_1A_2} = \sqrt{(A_1-A_2)^2}$

$$\Rightarrow \boxed{A_{3,min} = A_1 - A_2} \quad (\text{since } A_1 \geq A_2)$$

b) A_3 is maximum when $\cos(\phi_1-\phi_2)$ is max, i.e:

$\boxed{\cos(\phi_1-\phi_2) = 1}$, we need $\boxed{\phi_1-\phi_2 = 2n\pi,\ n\in\mathbb{Z}}$

and $A_{3,max} = \sqrt{A_1^2 + A_2^2 + 2A_1A_2} = \sqrt{(A_1+A_2)^2}$

$$\Rightarrow \boxed{A_{3,max} = A_1 + A_2} \quad (\text{since } A_1, A_2 \geq 0)$$