Report: Signals and Systems Lab Assignment 1

Bilkent University Electrical and Electronics Department
EE321-01

Semih Akkoç - 22103132

September 29, 2023

Part 1:

This part contains some Matlab exercises. The answers to those questions will be given accordingly,

(a) The difference between **a=[3.2 34/7 -6 24**] and **a=[3.2; 34/7; -6; 24**] is these vectors have different dimensions first one has 1x4 and the second one has 4x1 thus ';' operator separates numbers with rows.

(b) By writing **b=[3.2;34/7;-6;24];** it didn't overwrite the variable **a** compared to the part a.

(c) When a ';' is added at the end of the statement, it does not print the result of the statement in the command window; thus, putting a ';' at the end of the statement makes the code faster. However, if a variable wants to be displayed, it is better not to use ';'. The measured times are 1.8850e-04s for without ';' and 1.4250e-05s with the ';'.

(d) It gives an error message that points out that the '*' symbol is reserved for matrix multiplication due to given vector dimensions not being satisfied for matrix multiplication. It suggests whether I want to perform '.*' for element-wise multiplication.

(e) The result is **c=[18.56 23.3143 -30 -2448**]. As mentioned in the previous part, the effect of adding a dot in front of the '*' made the operation of element-wise multiplication. Writing **c=b.*a** did not change the result due to the commutative property of multiplication.

(f) The result is -2.4361e+03, and Matlab performed a matrix multiplication, and the output was a scalar since matrix multiplication was done on (1x4)*(4x1)=(1x1).

(g) The result was **c = [18.56 15.36 16 -326.4; 28.17 23.31 24.29 -495.43; -34.8 -28.8 -30 612; 139.2 115.2 120 -2448]**. and Matlab performed a matrix multiplication, and the output was a matrix since matrix multiplication was done on (4x1)*(1x4)=(4x4).

(h) It creates a vector that has elements starting from 1 to 2 with 0.01 increments, which results in a 1x101 dimensional vector.

(i) When ';' it added the time took for generation was 1.5458e-05s otherwise it was 6.0633e-04s.

(j) With generating the same vector using for loop took 0.0173s.

(k) With allocated memory, it took 0.0062s to generate the same vector; thus using, creating the vector with **a=[1:0.01:2]** is the most efficient method.

(l) Matlab applies the sin function to each element of **a** so it creates a new vector called **b**.

(m) When **plot(x)** was called, the x-axis was the indexes of the **x** vector, and the y-axis was the values of the indexes. **plot(x,t)** had t values in the y-axis and x values in the x-axis. **plot(x,t)** had x values in the y-axis and t values in the x-axis.

(n) Adding '-+' argument changes the line of the plot and adds '+' sign to each value in x. Changing the argument to '+' changes the plot, and it only marks the values of x and does not draw a line between consecutive points. They are drawn as discrete points.

(o) There are 26 time points included in **t**.

(p) Writing **t=linspace(1,4,151)** generates the same vector with the part m.

(q) While computing the $x(t)$ the t was **t=[0:0.04:1]** and required $x(t)$ has been computed.

(r) The $x(t)$ has been plotted as following,

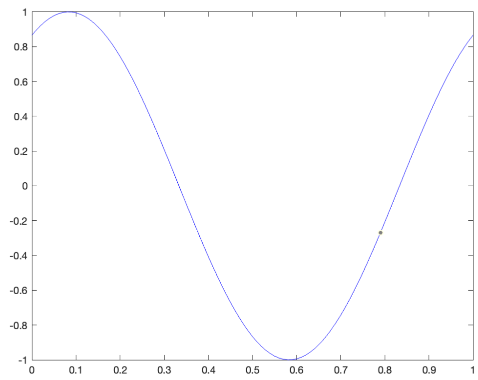

Figure 1: The plot of $x(t)$.

(s) After changing the **t** to **t=[0:0.01:1]** the plot looks like following,

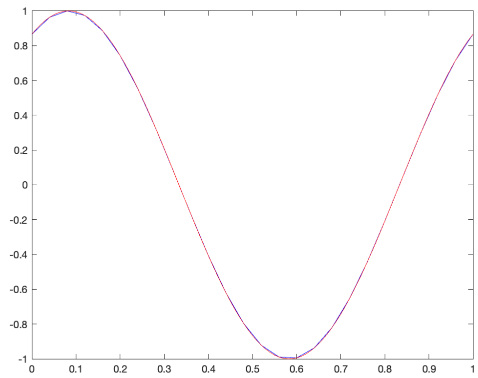

Figure 2: The plot of $x(t)$ with 2 different step-sized t's.

(t) Adding three **t** and changing the **t** to **t=[0:0.1:1]** the plot looks like following,

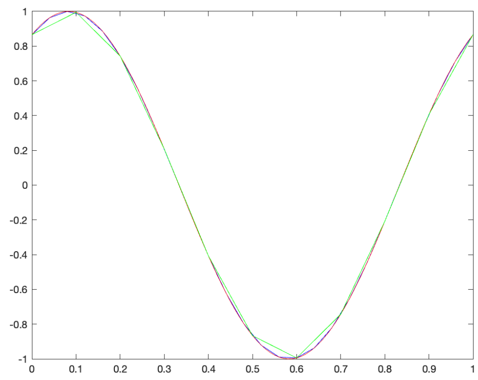

Figure 3: The plot of $x(t)$ with 3 different step-sized t's.

(u) Adding four **t** and changing the **t** to **t=[0:0.2:1]** the plot looks like following,

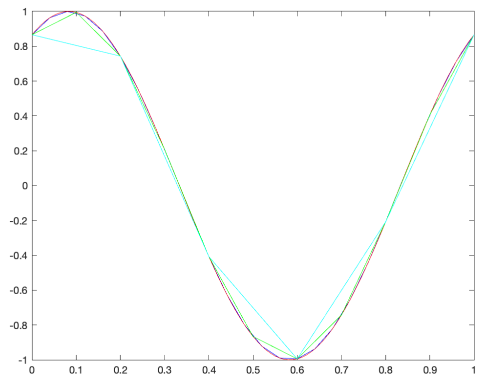

Figure 4: The plot of $x(t)$ with 4 different step-sized t's.

(v) After inspection, it is clear that the smaller the step size, the more continuous the $x(t)$ becomes. Therefore, between four different step-sized **t**'s most likely continuous would be generated with **t=[0:0.01:1]**.

(w) According to the documentation provided by Matlab's help function, if line style is not specified in the plot function, the plot uses a solid line. To do this, the plot function connects the consecutive data points with each other.

(x) The difference between the stem function and the plot function is that the stem function plots the data points discretely, and it would more likely be more useful with discrete point plotting.

Part 2:

This part contains how different signal waveforms sound.

1. (a) Difference between **sound** and **soundsc** is that first one expects values between -1 and 1 else it clips the value. The second one does not clip the values therefore, data is scaled compared to **sound** function. They are both appropriate to listen to the given function.

   (b) The plot of the of $f_0 = 440$ can be seen as following,

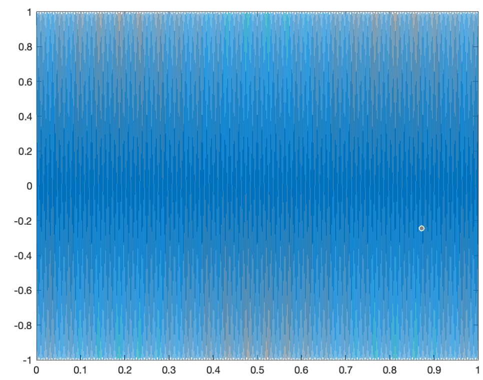

Figure 5: The plot of $x(t)$ when $f_0 = 440$.

(c) The process has been repeated for $f_0 = 687$.

(d) The process has been repeated for $f_0 = 883$. As the frequency increases the pitch of the sound increases as well.

2. The code for obtaining **x2** can be found in following lines:

```
t=[0:1/8192:1];
f_0 = 330;
a=7;
x2=cos(2*pi*f_0*t).*exp(-a*t);
plot(t, x2);
sound(x2);
```

Plot of the **x2** is shown in the figure below:

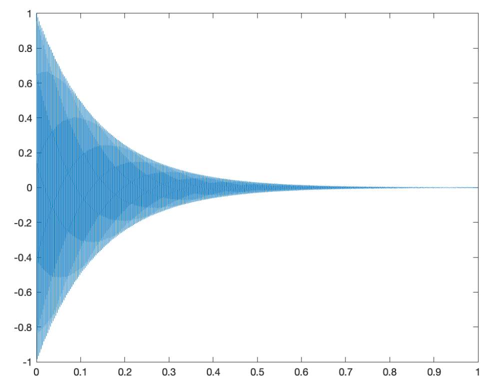

Figure 6: The plot of $x2$ when $f_0 = 330$ and $a = 7$.

Compared to the **x1**, the **x2** faded away slowly in contrast to the other one. This fading away is caused by the e^{-at} term. The **x2** sounded more like a piano, and the **x1** sounded more like a flute. As it increased, the sound more sharply ended therefore, the graph had greater decay.

3. The code for obtaining **x3** can be found in following lines:

```
t=[0:1/8192:1];
f_0 = 510;
f_1 = 4;
a=7;
x3=cos(2*pi*f_0*t).*cos(2*pi*f_1*t);
plot(t, x3);
sound(x3);
```

Plot of the **x3** is shown in the figure below:

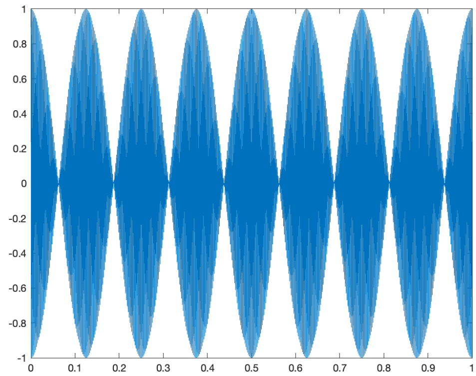

Figure 7: The plot of **x3** when $f_0 = 510$ and $f_1 = 4$.

Adding a low-frequency term changed the sound profile so that it had ups and downs in one run, unlike adding a decay like e^{-at}. As the f_1 increased, the period of the signal got smaller, and it had more ups and downs, and vice versa as the f_1 got smaller, its period got greater, and it had fewer ups and downs.

$$cos(2\pi f_0 t)cos(2\pi f_1 t) = cos(2\pi f_0 t)cos(2\pi f_1 t) + sin(2\pi f_0 t)sin(2\pi f_1 t) - sin(2\pi f_0 t)sin(2\pi f_1 t) \quad (1)$$

$$\Rightarrow 2cos(2\pi f_0 t)cos(2\pi f_1 t) = cos(2\pi f_0 t)cos(2\pi f_1 t)+sin(2\pi f_0 t)sin(2\pi f_1 t) - sin(2\pi f_0 t)sin(2\pi f_1 t) + cos(2\pi f_0 t)cos(2\pi f_1 t) \quad (2)$$

$$\Rightarrow 2cos(2\pi f_0 t)cos(2\pi f_1 t) = cos(2\pi(f_0 - f_1)t) + cos(2\pi(f_1 + f_0)t) \quad (3)$$

$$\Rightarrow cos(2\pi f_0 t)cos(2\pi f_1 t) = \frac{cos(2\pi(f_0 - f_1)t) + cos(2\pi(f_0 + f_1)t)}{2} \quad (4)$$

Part 3:

This part contains the instantaneous frequency concept and some mathematical operations to show instantaneous frequency.
Now it will be shown that $f_{ins}(t) = f_0$ for $x_1(t)$,

$$x_1(t) = cos(2\pi f_0 t) = cos(2\pi\phi(t)) \Rightarrow \phi(t) = f_0 t \Rightarrow f_{ins}(t) = \frac{d\phi(t)}{dt} = f_0$$

For $x_4(t)$ it will shown that $f_{ins}(t) = \alpha t$ for all t,

$$x_4(t) = cos(\pi\alpha t^2) = cos(2\pi\phi(t)) \Rightarrow \phi(t) = \frac{\alpha t^2}{2} \Rightarrow f_{ins}(t) = \frac{d\phi(t)}{dt} = \frac{2\alpha t}{2} = \alpha t$$

The instantaneous frequency of $x_4(t)$ at $t = 0$ is 0 and at $t = t_0$ is αt_0. After generating a random number $\alpha = 1897$ the frequency values will change between 0 Hz and 1897 Hz. The code for generating **x4** and the plot of it is in the below figure,

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

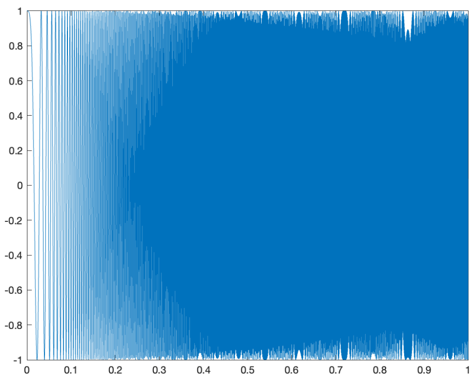

Figure 8: The plot of **x4** when $\alpha = 1897$.

As α increased signals frequency increased more rapidly therefore it sounded more faster than others.

Instantaneous frequency of $x_5(t)$ can be found with following equalities,

$$x_5(t) = cos(2\pi(-500t^2 + 1600t)) = cos(2\pi\phi(t)) \Rightarrow \phi(t) = -500t^2 + 1600t$$
$$\Rightarrow f_{ins}(t) = \frac{d\phi(t)}{dt} = -1000t + 1600$$

According to the previous calculations frequency at $t = 0$ is $f = 1600$ Hz and $t = 1$ is $f = 600$ Hz and at $t = 2$ is $f = -400$ Hz. As time passes, frequency decreases in linear fashion for **x5**. The code for **x5** is shown in the below lines.

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

Part 4:

When $x(t) = cos(2\pi\alpha t + \phi)$ and as ϕ increases from 0 to π it is noticeable that the volume increases. However, the pitch doesn't change. Since by changing ϕ, the signal is shifted to the left as it is increased but the frequency stays the same.

Part 5:

This part will include the results of $x_3(t) = x_1(t) + x_2(t)$ where $x_1(t) = A_1cos(2\pi f_0t + \phi_1)$ and $x_2(t) = A_2cos(2\pi f_0t + \phi_2)$.

$$x_3(t) = A_1cos(2\pi f_0t + \phi_1) + A_2cos(2\pi f_0t + \phi_2) \tag{1}$$

$$\begin{aligned} x_3(t) = A_1[cos(2\pi f_0t)cos(\phi_1) - sin(2\pi f_0t)sin(\phi_1)] \\ + A_2[cos(2\pi f_0t)cos(\phi_2) - sin(2\pi f_0t)sin(\phi_2)] \end{aligned} \tag{2}$$

$$\begin{aligned} x_3(t) = [A_1cos(\phi_1) + A_2cos(\phi_2)]cos(2\pi f_0t) \\ - [A_1sin(\phi_1) + A_2sin(\phi_2)]sin(2\pi f_0t) \end{aligned} \tag{3}$$

$$\begin{aligned} x_3(t) = (A_1cos(\phi_1) + A_2cos(\phi_2))[cos(2\pi f_0t) \\ - sin(2\pi f_0t)\frac{A_1sin(\phi_1) + A_2sin(\phi_2)}{A_1cos(\phi_1) + A_2cos(\phi_2)}] \end{aligned} \tag{4}$$

Let $tan(\phi_3) = \frac{A_1 sin(\phi_1)+A_2 sin(\phi_2)}{A_1 cos(\phi_1)+A_2 cos(\phi_2)}$ then,

$$x_3(t) = (A_1 cos(\phi_1) + A_2 cos(\phi_2))[cos(2\pi f_0 t) - sin(2\pi f_0 t) tan(\phi_3)] \tag{5}$$

$$x_3(t) = \frac{A_1 cos(\phi_1) + A_2 cos(\phi_2)}{cos(\phi_3)}(cos(2\pi f_0 t) cos(\phi_3) - sin(2\pi f_0 t) sin(\phi_3)) \tag{6}$$

$$x_3(t) = \frac{A_1 cos(\phi_1) + A_2 cos(\phi_2)}{cos(\phi_3)} cos(2\pi f_0 t + \phi_3) \tag{7}$$

Let $A_3 = \frac{A_1 cos(\phi_1)+A_2 cos(\phi_2)}{cos(\phi_3)}$ then,

$$x_3(t) = A_3 cos(2\pi f_0 t + \phi_3) \tag{8}$$

So, A_3 and ϕ_3 can be written in terms of A_1, A_2, ϕ_1, and ϕ_2 as following:

$$tan(\phi_3) = \frac{A_1 sin(\phi_1) + A_2 sin(\phi_2)}{A_1 cos(\phi_1) + A_2 cos(\phi_2)}$$

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

And ϕ_3 as:

$$A_3 = \frac{A_1 cos(\phi_1) + A_2 cos(\phi_2)}{cos(\phi_3)}$$

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

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

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

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

So for $\phi_1 - \phi_2 = n\pi$ where $n = ..., -3, -1, 1, 3, ...$ odd numbers the A_3 is the smallest and for $n = ..., -2, 0, 2, ...$ even numbers A_3 gets its maximum value. Which gives as for $\max(A_3) = \sqrt{{A_1}^2 + {A_2}^2 + A_1 A_2}$ and for minimum value $\min(A_3) = \sqrt{{A_1}^2 + {A_2}^2 - A_1 A_2}$.