ECE 340 Lab 1

Omar Mahmoud
1753607
Section D21

9/25/2024

1 Signal Generation and Plotting

Consider the following discrete functions:

$$x_1[k] = -5.1\sin(0.1\pi k - 3\pi/4) \quad -10 \le k \le 10$$

$$x_2[k] = -(0.9)^k e^{(j\pi k/10)} \quad 0 \le k \le 100$$

To start, $x_1[k]$ can be plotted on MATLAB through the following script:

```
subplot(2, 2, [1 2]);
%% Define x_1
k = -10:40;
x1 = -5.1*sin(0.1*pi.*k - 3*pi/4) + 1.1*cos(0.4*pi.*k);
%% Plot x_1
stem(k, x1, 'LineWidth', 1, 'Color', 'b');
title("Plot of x1[k].");
xlabel("k");
ylabel("x1[k]");
```

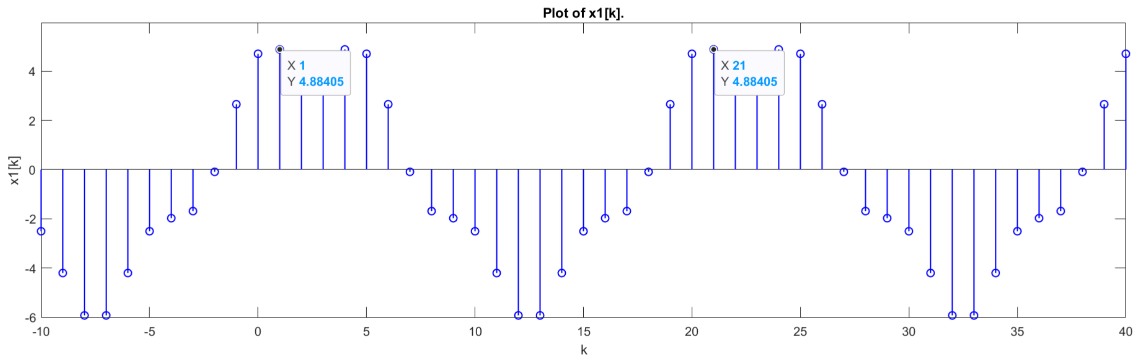

Figure 1: Plot of $x_1[k]$ with two points labeled to find the period.

By examining the graph, it is clear that $x_1[k]$ is a periodic sequence. To find the period, subtract the X of the two selected points:

$$T_{x_1} = X_2 - X_1 = 21 - 1 = 20$$

When plotting x_2 an imaginary and the real component will be produced. These can be plotted seperately through the following script:

```
%% Define x_2
k = 0:100;
x2 = (-0.9).^k .* exp(1).^(1j*pi.*k./10);
%% Plot the real component of the x_2
subplot(2, 2, 3);
stem(k, real(x2),'LineWidth', 1, 'Color', 'r');
title("Real Plot of x2[k].");
xlabel("k");
ylabel("Re(x2[k])");
%% Plot the img component of x_2
subplot(2, 2, 4);
stem(k, imag(x2),'LineWidth', 1, 'Color', 'r');
title("Imaginary Plot of x2[k].");
xlabel("k");
ylabel("Im(x2[k])");
```

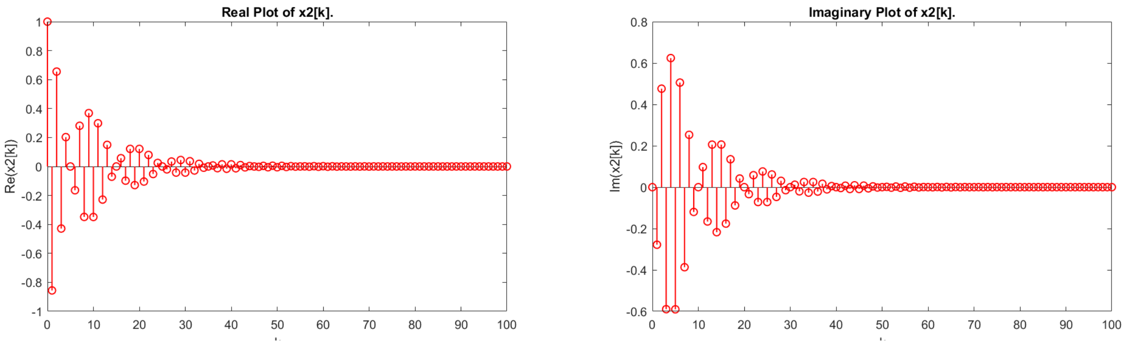

Figure 2: Plots of $Re(x_2[k])$ and $Im(x_2[k])$.

It is clear from the plots that $x_2[k]$ is not periodic, therefore it is not necessary to find a period.

To find the total energy of $x_1[k]$ and $x_2[k]$ we employ the following equation for discrete-time signals:

$$E_x = \sum_{n=-N}^{N} |x[n]|^2$$

where E_x represents the total energy of the signal over $[-N, N]$. The energy can be computed using the MATLAB script provided below:

```
%% Get energy of x_1
fprintf('E_x1 = %.4f\n', sum(abs(x1).^2));
%% Get energy of x_2
fprintf('E_x2 = %.4f\n', sum(abs(x2).^2));
```

The output of the script is as follows:

```
E_x1 = 698.5335
E_x2 = 5.2632
```

2 Digital Audio

MATLAB provides features that facilitate the reading and writing of audio signals from and to .wav files. The following script reads the audio file **baila.wav**, and stores its audio signal into the matrix, x_3 along with its sampling rate in F_s. In this context, the rows of x_3 represents the samples, while the columns represent the channels of the audio signal:

```
%% Read the audio file baila.wav
[x3, Fs] = audioread('baila.wav');
[num_samples, num_channels] =  size(x3);
disp(['Number of samples: ', num2str(num_samples)]);
disp(['Number of channels: ', num2str(num_channels)]);
```

The output of the script is as follows:

```
Number of samples: 1000000
Number of channels: 1
```

From the output, it is evident **baila.wav** contains a mono-channel signal with 1,000,000 samples. To analyze this signal, the next step involves creating an apporpriate time vector and plotting the mono signal against it using the following script:

```
%% Create time vector
t = (0:num_samples-1) / Fs;
%% Plot the audio signal against it
figure(1);
plot(t, x3);
ylabel('Amplitude');
xlabel('Time (seconds)');
title('Audio Signal over Time');
```

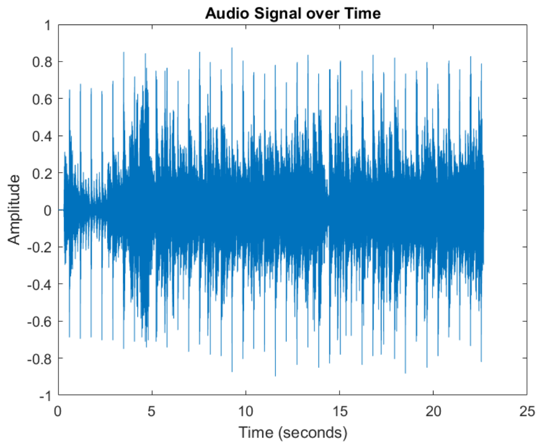

Figure 3: Plot of the mono audio signal x_3 extracted from **baila.wav**.

To assess the total energy of x_3 we employ the same method as in Section 1:

```
%% Get total energy
fprintf('E_x3 = %.4f\n', sum(abs(x3).^2));
```

The value generated by the script is as follows:

```
E_x3 = 20422.7731
```

To showcase MATLAB's audio writing features, the following script samples the first half of the matrix x_3 and creates a new matrix x_{3s}. This matrix is then used to play half of the **baila.wav** file using MATLAB's *sound* function and generates a new **baila_half.wav** file using the *audiowrite* function:

```
%% Get x3s, play it, and create baila_half.wav
x3s = x3(1:floor(num_samples/2), :);
sound(x3s, Fs);
audiowrite('baila_half.wav', x3s, Fs);
```