OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | EIT | FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIONSTECHNIK

Laboratory

Digital Information Processing

Report

Names Harish Kongari [Mr. Nr. 214586]
Harikrishna Polavarapu [Mr. Nr. 218060]

Experiment Digital signal processing in Matlab

Date 26/04/2018

Contents

1 Content of the Experiment

In this experiment you will get to know the bases of Matlab and how it can be used for signal processing. You will also learn different kinds of transformation and why they are useful.

2 Preparatory Exercises

Exercise 2.1: Why transformations?

What is the purpose of transformations like e.g. the z-transform?

Answer

Transformations are generally used to simplify the complex representation of a signal and system thus enables us to understand better and also we can modify it in a less complex representation. [1

e.g., z-transform:

- Z-transform is a powerful tool used in analysis of signals and linear Time Invariant (LTI) Systems. [1]
- Is used to simplify discrete time signals. e.g., digital signal processing, digital filter design.
- Simplifies the analysis of the response of an LTI system.
- Characterizations like stability, causality can be determined easily.
- Pole-Zero analysis is one more advantage.

Exercise 2.2: Computation of a z-transform

Compute the z-transform of the series $x(n) = a^{|n|}, -1 < a < 1$.

Answer

x (n) = a^n u[n] + a^{-n} u[-n-1]

Applying z-transform,

Z (x (n)) = X (z) = $\sum_{n=-\infty}^{\infty} a^n u[n]\, z^{-n} + \sum_{n=-\infty}^{\infty} a^{-n} u[-n-1]\, z^{-n}$

$= \sum_{n=0}^{\infty} (az^{-1})^n + \sum_{n=-\infty}^{-1} (a^{-1}z^{-1})^n$

$= \sum_{n=0}^{\infty} (az^{-1})^n + \sum_{n=1}^{\infty} (az)^n$

$= \frac{1}{1-az^{-1}} + (-1+1+\sum_{n=1}^{\infty} (az)^n)$ by using Taylor series expansion.

$= \frac{1}{1-az^{-1}} + (-1+\sum_{n=0}^{\infty} (az)^n)$

$= \frac{1}{1-az^{-1}} + (-1+\frac{1}{1-az})$

$= \frac{1}{1-az^{-1}} + (\frac{az}{1-az})$

$= \frac{z}{z-a} + \frac{az}{1-az}$

Therefore X (z) $= \frac{z}{z-a} - \frac{z}{z-(\frac{1}{a})}$ ---------- (1)

Exercise 2.3: From the z-transform to the Fourier transform

Use the z-transform of the series to derive its Fourier transform: $X(\omega) = \frac{1-a^2}{1-2a\cos(\omega)+a^2}$

Answer

If a fourier transform exists for x(n), then Region of convergence (ROC) includes unit circle. Hence we could substitute z = $e^{j\omega}$ in above eq. (1).

$$X(\omega) = \frac{e^{j\omega}}{e^{j\omega}-a} - \frac{e^{j\omega}}{e^{j\omega}-(\frac{1}{a})} = \frac{-e^{j\omega}.\frac{1}{a}+e^{j\omega}.a}{e^{2j\omega}-e^{j\omega}.\left(a+\frac{1}{a}\right)+1} = \frac{-1+a^2}{a.e^{j\omega}-a^2-1+a.e^{-j\omega}} = \frac{1-a^2}{a^2+1-a.(e^{j\omega}+e^{-j\omega})}$$

$$= \frac{1-a^2}{a^2+1-2.a.\cos(\omega)} \qquad \text{Since } \cos(\omega) = \frac{e^{j\omega}+e^{-j\omega}}{2}$$

Exercise 2.4: Why MATLAB

Mention some advantages of Matlab.

Answer

Advantages of MATLAB [2]:

1. The vast toolboxes it provides, helps one work in different fields of science.
2. Many built-in functions helps the user to decrease the complexity of the program.
3. IDE (Integrated Development Environment) is already included in MATLAB.
4. Errors are easy to fix with the help of its debugging techniques by applying breakpoints.
5. MATLAB Documentation supports users in getting all the help needed in its operation.
6. It has built in graphics for data visualization and has tools for 2D and 3D plots.
7. MATLAB is used as a calculator, with features faster complex calculations.

Exercise 2.5: Drawing with Matlab

Use Matlab to define a vector of length 10000. Its values should be between 0 and 2π. Draw the sine-function of that vector using the command „plot“.

Matlab Code:

```
x = linspace(0,2*pi,10000);   % Creating a vector of length 10000 between 0 and 2*pi
y = sin(x);                   % Helps in knowing sine value at each value of vector
figure(1);                    % Opens figure window
plot(x,y)                     % Plotting graph between x and y values
hold on                       % Continues to add other commands on the same figure
grid on                       % Turns ON grid lines
xlabel('x','color','r');      % Labelling the x-axis
ylabel('sin(x)','color','r'); % Labelling the y-axis
title('y = sin(x)')           % Adding title to the plot
```

In the above script code, we defined a vector of length 10000 for ‘x’ between [0, 2*pi] and variable ‘y’ is assigned with sin(x), which carries the sine value at 10000 values of ‘x’. The function ‘plot’ helps in plotting the sine-function of ‘y’ with respect to the values of ‘x’. The corresponding sine function of the vector values is as shown in *Figure 2.5* below.

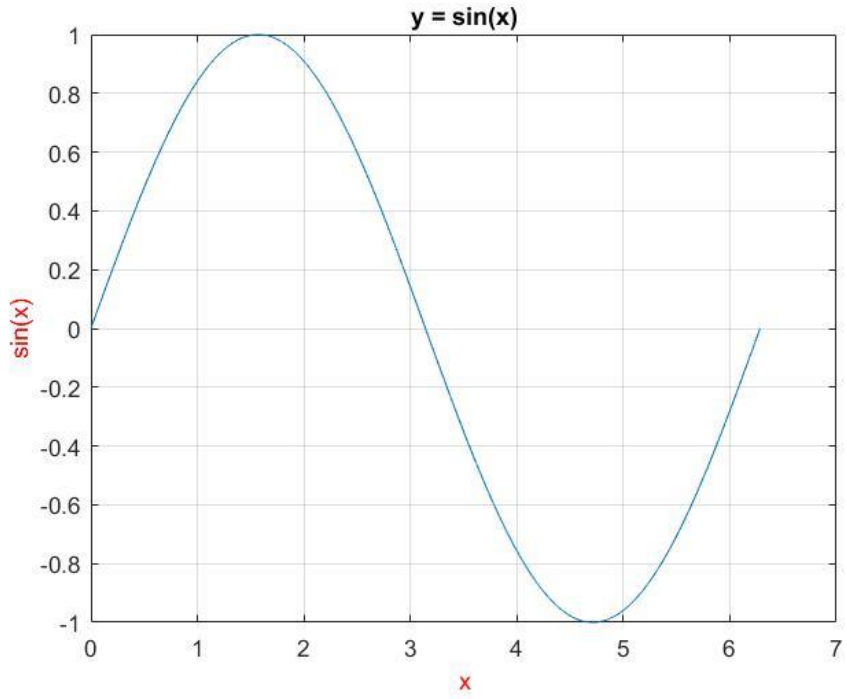

Figure 2.5: Sine-function of vector of length 10000.

Exercise 2.6: Sampling

Sample the sine by taking only one-hundredth of the values and draw the sampled sine using the function "stem".

Matlab Code:

```
x = linspace(0,2*pi,100);      % Creating a vector of length 100 between 0 and 2*pi
y = sin(x);                    % Helps in knowing sine value at each value of vector
figure(1);                     % Opens figure window
stem(x,y)                      % Plotting discrete sequence data
hold on                        % Continues to add other commands on the same figure
grid on                        % Turns ON grid lines
xlabel('x','color','r');       % Labelling the x-axis
ylabel('sin(x)','color','r');  % Labelling the y-axis
title('y = sin(x)')            % Adding title to the plot
```

In the above script code, we divided 'x' into 100 parts (same as exercise 2.5 but instead of 10000 here we took only 100 values of the vector length) between [0, 2*pi], variable 'y' is assigned with sin(x), which carries the sine value of 100 values of 'x'. Function 'stem' helps in plotting 100 discrete values of 'y' at respective 100 discrete values of 'x'. The corresponding sampled sine function using *"stem"* function of the vector values is as shown in *Figure 2.6* below.

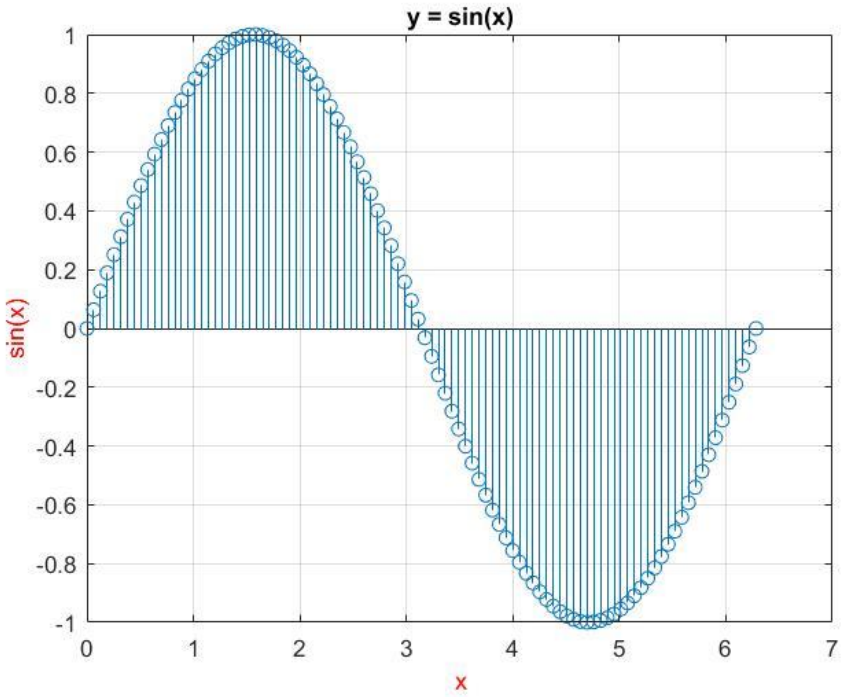

Figure 2.6: Sampled sine-function using the "stem" function.

3 Exercises with Results

Given is:

Given is the real valued series $x(n) = a^{|n|}$ for $-1 < a < 1$. The Fourier transform of this series is:

$$X(\omega) = \frac{1 - a^2}{1 - 2a\cos(\omega) + a^2}$$

Exercise 3.1: Graphical illustration

Plot the series $x(n)$ and its spectrum $X(\omega)$ for $a = 0{,}8$.

Hint: Use the function „stem“ for the series and the function „plot“ for the spectrum.

Matlab Code:

```
function series_spectrum(a)                    % Defining a function with input 'a'
n = -100:1:100;                                % Range of the integer interval
x_n = a.^(abs(n));                             % Defining the real valued series
figure(1)                                      % Opening the figure window
stem(n,x_n);                                   % Plotting discrete sequence points
hold on                                        % Holding the figure for further changes
grid on                                        % To add grid lines in the figure plot
xlabel('n', 'color','r');                      % Labelling the x-axis
ylabel('x(n)','color','r');                    % Labelling the y-axis
title('Real valued series x(n)','color','r');  % Giving title to the figure
hold off                                       % Turning OFF hold on function
w= linspace(0,2*pi,201);                       % Dividing 201 parts between 0 and 2*pi
X_w = (1 - a^2)./(1- 2*a*cos(w) + a^2);        % Defining the spectrum
figure(2)                                      % Opening the figure window
plot(w,X_w)                                    % Plot corresponding spectrum
hold on                                        % Holding the figure for further changes
grid on                                        % To add grid lines in the figure plot
xlabel('w', 'color', 'r')                      % Labelling the x-axis
ylabel('X(w)','color','r')                     % Labelling the y-axis
title('Spectrum X(w)','color','r')             % Giving title to the figure
hold off                                       % Turning OFF hold on function
end                                            % Ends the function
```

In the above code, function ‘series_spectrum(a)’ has a single input variable ‘a’ and it is expected to be in the range (-1,1) i.e., $-1 < a < 1$.

The function ‘series_spectrum(a)’ give us visual explanation of the real value series ‘x(n)’ and the spectrum ‘X(w)’ of the real value series.

Here, we have taken the interval range of the real value series x(n) as [-100,100] which results in increase in 1 unit from -100 to 100 . So, total 201 points are taken into consideration. The function *“stem”* helps in plotting 201 discrete points of ‘n’ for series ‘x(n)’.The corresponding plot is as shown in *Figure 3.1.1* below.

Similarly the Spectrum ‘X(w)’ in interval [0, 2*pi] was divided into 201 parts with the help of the ‘linspace’ function and with the ‘plot’ function we plot the Spectrum in the interval. The corresponding plot is as shown in *Figure 3.1.2* below.

Calling the function in command window, series_spectrum(0.8);

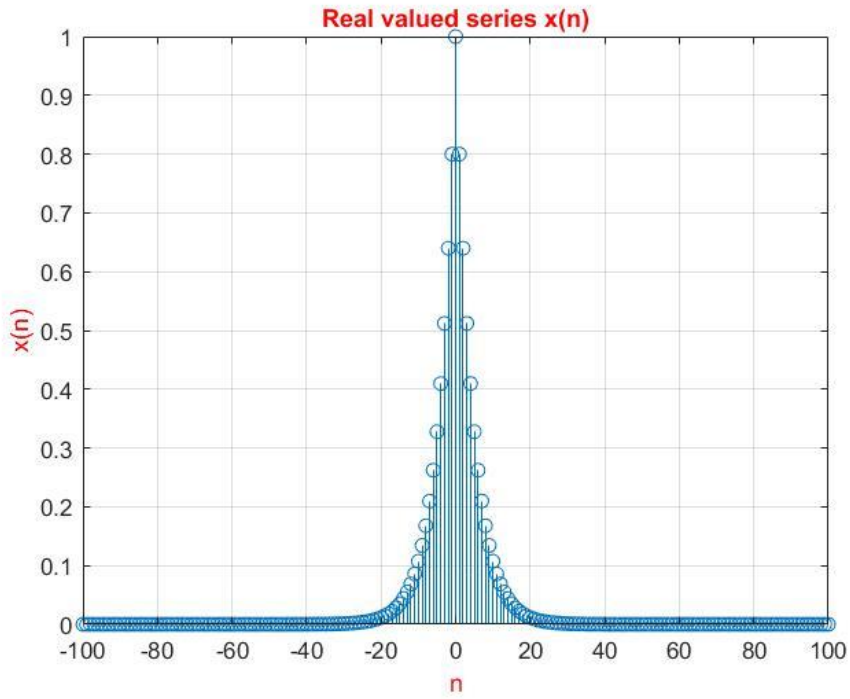

Figure 3.1.1: Plot of the real valued series x(n) using the“stem” function .

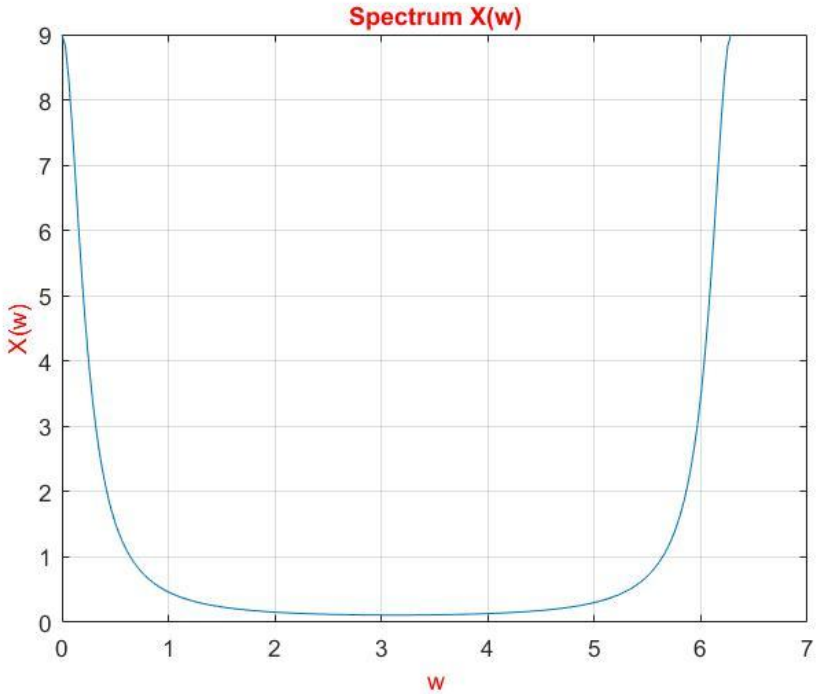

Figure 3.1.2: Plot of the spectrum X(w) for a = 0.8.

Exercise 3.2: Windowing of the series

Sample $x(n)$ in the integer interval $[-10,10]$. Plot the resulting series $x_{10}(n)$with „stem“.

Matlab Code:

```
function series(a)                                  % Defining a function with input 'a'
n = -10:10;                                         % Range of the integer interval
x_n = a.^(abs(n));                                  % Defining the real valued series
figure(1)                                           % Opening the figure window
stem(n,x_n);                                        % Plotting discrete sequence points
hold on                                             % Holding the figure for further changes
grid on                                             % To add grid lines in the figure plot
xlabel('n','color','r');                            % Labelling the x-axis
ylabel('x_{10}(n)','color','r');                    % Labelling the y-axis
title('Resulting series x_{10}(n)','color','r');    % Giving title to the figure
hold off                                            % Turning OFF hold on function
end                                                 % Ends the function
```

In the above code, function ‘series(a)’ has a single input variable ‘a’ and it is expected to be in the range (-1,1) i.e., $-1 < a < 1$.

The function ‘series(a)’ give us visual explanation of the real value series ‘x_{10}(n)’.Here, we have taken the interval range of the real value series ‘x(n)’ as [-10, 10] which results in increase in 1 unit from -10 to 10 .So, total 21 points are taken into consideration. The function *“stem”* helps in plotting 21 discrete points of ‘n’ for series ‘x(n)’ i.e., Resulting series ‘x_{10}(n)’.The corresponding plot is as shown in *Figure 3.2* below.

Calling the function in command window, series(0.8);

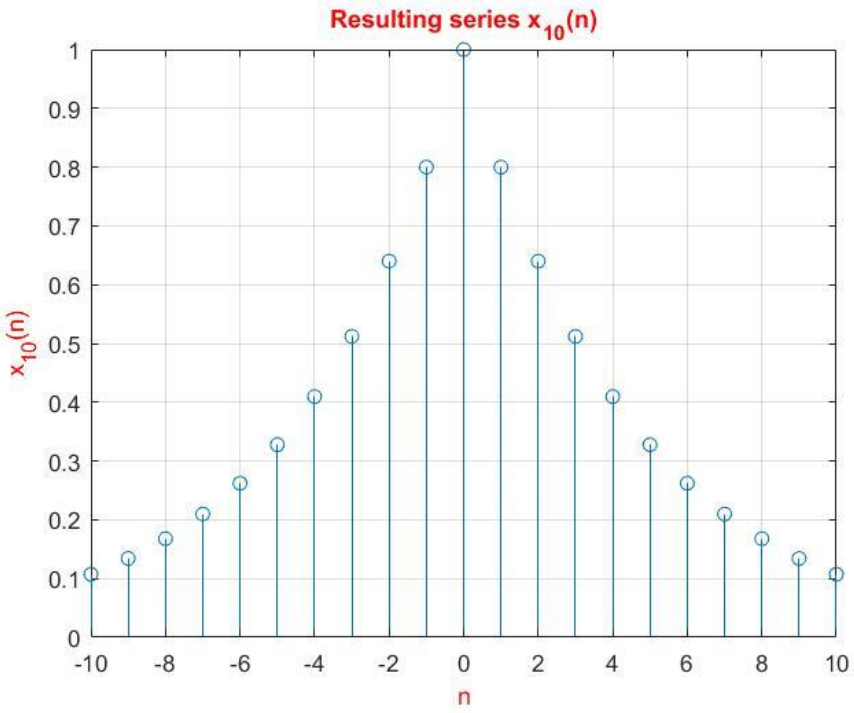

Figure 3.2: Plot of the series $x_{10}(n)$ using the “stem” function.

Exercise 3.3: Spectrum of the windowed series

Compute the DFT (Matlab command: „fft“) of the series $x_{10}(n)$. Then plot the spectrum $X_{10}(\omega)$in the real-valued interval $[0,2\pi]$using „plot“.

Matlab Code:

```
function series2spectrum(a)                    % Defining a function with input 'a'
n = -10:1:10;                                  % Range of the integer interval
x_n = a.^(abs(n));                             % Defining the real valued series
X_w = abs(fft(x_n));                           % Applying 'fft' to the x_n
w = linspace(0,2*pi,length(n));                % Dividing into parts of length n between 0-2*pi
figure(1)                                      % Opening the figure window
plot(w,X_w);                                   % Plot corresponding spectrum
hold on                                        % Holding the figure for further changes
grid on                                        % To add grid lines in the figure plot
xlabel('w','color','r');                       % Labelling the x-axis
ylabel('X_{10}(w)','color','r');               % Labelling the y-axis
title('Spectrum of X_{10}(w)','color','r');    % Giving title to the figure
hold off                                       % Turning OFF hold on function
end                                            % Ends the function
```

***Note**: The function 'fft' gives the complex values, so to avoid the warning with the MATLAB code stating the Imaginary values are being omitted, so the function 'abs' was used to function the absolute value of the complex value, calculating the magnitude of the real and imaginary part, proper plotting is done.*

In the above code, function 'series2spectrum(a)' has a single input variable 'a' and it is expected to be in the range (-1,1) i.e., $-1 < a < 1$.

The function 'series2spectrum(a)' give us visual explanation of the spectrum 'X_{10}(w)' which is the DFT (Matlab command: „fft“) of series 'x_{10}(n)' in range [0,2*pi]. Here, we have taken the interval range of the real value series 'x(n)' as [-10, 10] which results in increase in 1 unit from -10 to 10 .So, total 21 points are taken into consideration. The DFT (Matlab command: „fft“) of the series $x_{10}(n)$ results in spectrum 'X_{10}(w)' in the real-valued interval $[0,2\pi]$ is as shown in *Figure 3.3* below.

Calling the function in command window, series2spectrum(0.8);

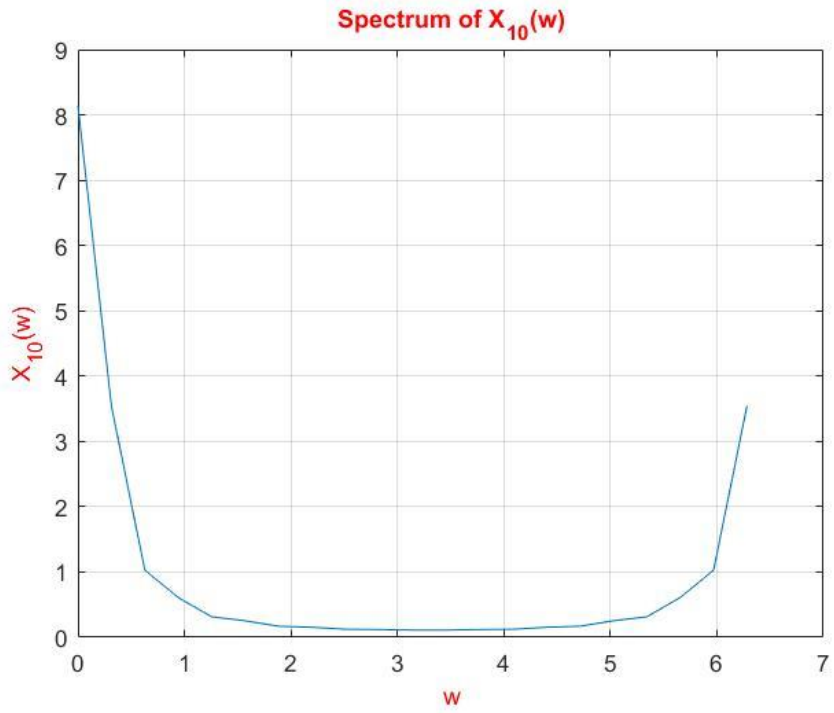

Figure 3.3: Plot of $X_{10}(w)$ in real value interval $[0, 2\pi]$.

Exercise 3.4: Comparison: ideal spectrum <=> spectrum of the windowed series

Plot the analytical determined Fourier transform $X(\omega) = \frac{1-a^2}{1-2a\cos(\omega)+a^2}$ and the spectrum $X_{10}(\omega)$ in the same figure (both in one graph) and compare them.

Matlab Code:

```
function compare_ideal_window(a)                % Defining a function with input 'a'
n1 = -100:1:100;                                % Range of the integer interval of X(w)
w1 = linspace(0,2*pi,length(n1));       % Dividing into parts of length n1 between 0-2*pi
n_10 = -10:10;                                  % Range of the integer interval of x_n_10
w_10 = linspace(0, 2*pi, length(n_10));% Dividing into parts of length n_10 between 0-2*pi
X_w = (1 - a^2)./(1- 2*a*cos(w1) + a^2);        % Defining the spectrum X(w)
figure(1)                                       % Opening the figure window
plot(w1,X_w)                                    % Plot corresponding spectrum X(w)
hold on                                         % Holding the figure for further changes
grid on                                         % To add grid lines in the figure plot
x_n_10= a.^(abs(n_10));                         % Defining the real valued series
X_w_10 = abs(fft(x_n_10));                      % Applying 'fft' to the x_n_10
plot(w_10,X_w_10);                              % Plot corresponding spectrum X10(w)
grid on                                         % To add grid lines in the figure plot
xlabel('Normalised frequency','color','r');     % Labelling the x-axis
ylabel('Magnitude','color','r');                % Labelling the y-axis
title('Ideal spectrum vs Spectrum of windowed series','color','r');    % Giving title to the figure
legend('X(w)','X_{10}(w)')                      % Adding legend to the figure
hold off                                        % Turning OFF hold on function
end                                             % Ends the function
```

***Note**: The function 'fft' gives the complex values, so to avoid the warning with the MATLAB code stating the Imaginary values are being omitted, so the function 'abs' was used to function the absolute value of the complex value, calculating the magnitude of the real and imaginary part, proper plotting is done.*

In the above code, function 'compare_ideal_window(a)' has a single input variable 'a' and it is expected to be in the range (-1,1) i.e., $-1 < a < 1$.

At first ideal spectrum 'X(w)' which is the fourier transform of 'x(n)'. The interval range for 'X(w)' is taken from [-100,100] .Later, spectrum of windowed series 'X_{10}(w)' which is the fourier transform of 'x_{10}(n)' we used interval range for 'X_{10}(w)' from [-10,10] .For comparison study 'X(w)' and 'X_{10}(w)' both are plotted in the same figure (both in one graph) as shown in the *Figure 3.4* below.

In theory according to sampling theorem or Nyquist criteria, the more samples we take the better those samples approximate the original function. A continuous band limited function can be completely represented by a set of equally spaced samples if the samples occur at more than twice the frequency of the highest frequency component of the function [3].

When we sample a signal, the sampling should be done according to Nyquist criteria in order to recover the original signal from the sampled signal without any loss of signal information.If this criteria is not met the original signal cannot be reconstructed. Here ideal spectrum X(w) is defined in interval [-100,100] and windowed signal X_{10}(w) between [-10,10], we are losing some signal data. Because of this we can observe the changes in plot.

Calling the function in command window, compare_ideal_window(0.8);

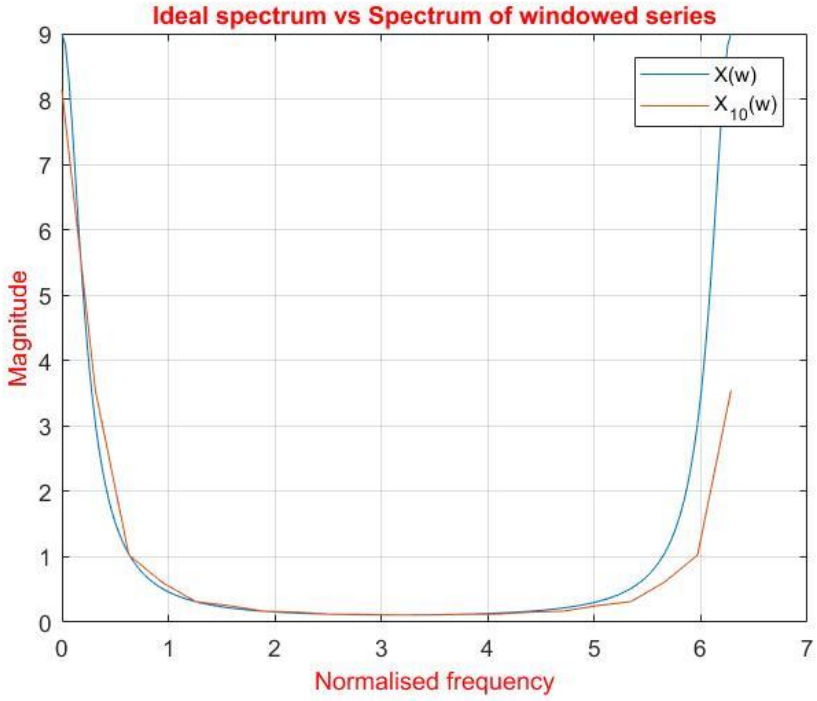

Figure 3.4: Comparison of Spectrums X(w) and X_{10}(w).

Exercise 3.5: Repeat for an expanded rectangular window

Repeat Exercise 3.2 to 3.4 for the integer interval $[-200,200]$and compare the results.

Answer

Here the basic idea is that we are implementing again from Exercise 3.2 to 3.4 with the integer interval [-200,200].The increase in integer interval is nothing but increasing window size. The increased window size in general results in the avoidance of the loss of signal data.

Matlab Code:

```
function expanded32compare(a)                    % Defining a function with input 'a'
subplot(2,1,1)                                   % Dividing the current plot into subplot
series(a);                                       % Calling function from Exercise 3.2
n = -200:200;                                    % Range of the integer interval
x_n = a.^(abs(n));                               % Defining the real valued series
subplot(2,1,2)                                   % Dividing the current plot into subplot
stem(n,x_n);                                     % Plotting discrete sequence points
hold on                                          % Holding the plot for further changes
grid on                                          % To add grid lines in the plot
xlabel('n', 'color','r');                        % Labelling the x-axis
ylabel('x_{200}(n)','color','r');                % Labelling the y-axis
title('Real Valued Series x_{200}(n)','color','r');      % Giving title to the plot
hold off                                         % Turning OFF hold on function
end                                              % Ends the function
```

In the above code, function 'expanded32compare(a)' has a single input variable 'a' and it is expected to be in the range (-1,1) i.e., $-1 < a < 1$. The function 'expanded32compare(a)' give us visual explanation of comparision of real value series '$x_{10}(n)$' and '$x_{200}(n)$'.

Here, we called series(a) from Exercise 3.2 which took the interval range of the real value series 'x(n)' as [-10, 10] which results in increase in 1 unit from -10 to 10 .So, total 21 points are taken into consideration. The function *"stem"* helps in plotting 21 discrete points of 'n' for series 'x(n)' i.e., Resulting series '$x_{10}(n)$'.

Later, we have taken the interval range of the real value series 'x(n)' as [-200,200] which results in increase in 1 unit from -200 to 200 .So, total 401 points are taken into consideration. The function *"stem"* helps in plotting 401 discrete points of 'n' for series 'x(n)' i.e., Real value series '$x_{200}(n)$'.The corresponding plots of '$x_{10}(n)$' and '$x_{200}(n)$' are as shown in *Figure 3.5.1* below. We can observe from the figure below that the series plot '$x_{200}(n)$' includes more discrete value points of the signal than the '$x_{10}(n)$'.

Calling the function in command window, expanded32compare(0.8);

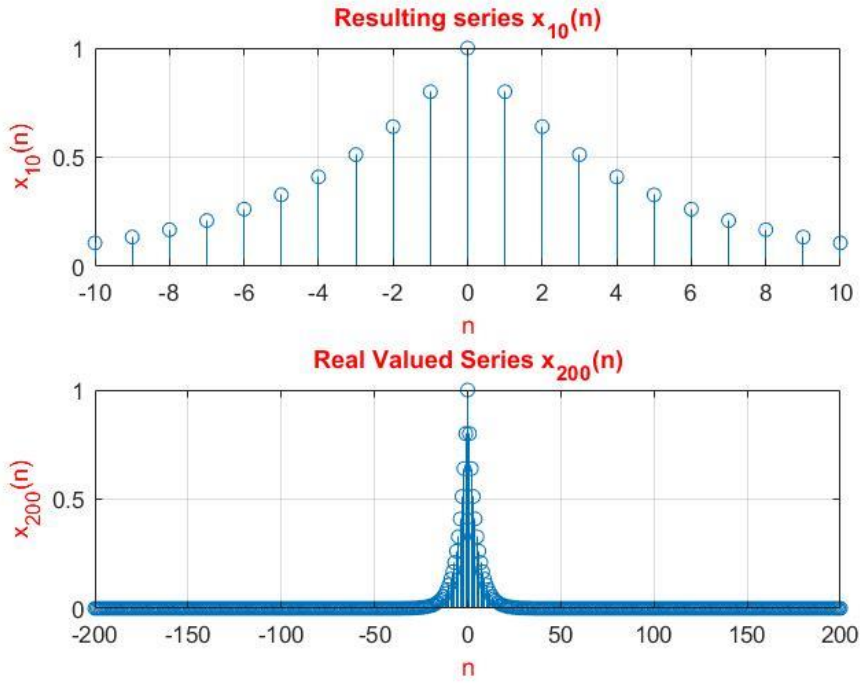

Figure 3.5.1: Comparison of real value series $x_{10}(n)$ and $x_{200}(n)$.

Matlab Code:

```
function expand33compare(a)                    % Defining a function with input 'a'
subplot(2,1,1)                                 % Dividing the current plot into subplot
series2spectrum(a);                            % Calling function from Exercise 3.3
n = -200:200;                                  % Range of the integer interval
x_n = a.^(abs(n));                             % Defining the real valued series
X_w = abs(fft(x_n));                           % Applying 'fft' to the x_n
w = linspace(0,2*pi,length(n));                % Dividing into parts of length n between 0-2*pi
subplot(2,1,2)                                 % Dividing the current plot into subplot
plot(w,X_w);                                   % Plot corresponding spectrum
hold on                                        % Holding the plot for further changes
grid on                                        % To add grid lines in the plot
xlabel('w','color','r');                       % Labelling the x-axis
ylabel('X_{200}(w)','color','r');              % Labelling the y-axis
title('Spectrum of X_{200}(w)','color','r');   % Giving title to the plot
hold off                                       % Turning OFF hold on function
end                                            % Ends the function
```

***Note**: The function 'fft' gives the complex values, so to avoid the warning with the MATLAB code stating the Imaginary values are being omitted, so the function 'abs' was used to function the absolute value of the complex value, calculating the magnitude of the real and imaginary part, proper plotting is done.*

In the above code, function ‘expand33compare(a)’ has a single input variable ‘a’ and it is expected to be in the range (-1,1) i.e., $-1 < a < 1$. The function ‘expand33compare(a)’ give us visual explanation of the spectrum ‘$X_{10}(w)$’ and ‘$X_{200}(w)$’.

Here, we called series2spectrum(a) from Exercise 3.3 which took the interval range of the real value series ‘x(n)’ as [-10, 10] which results in increase in 1 unit from -10 to 10. So, total 21 points are taken into consideration. The DFT (Matlab command: „fft“) of series ‘$x_{10}(n)$’ in range [0,2*pi] is used to find ‘$X_{10}(w)$’.

Similarly, we have taken the interval range of the real value series ‘x(n)’ as [-200,200] which results in increase in 1 unit from -200 to 200 . So, total 401 points are taken into consideration. The DFT (Matlab command: „fft“) of series ‘$x_{200}(n)$’ in range [0,2*pi] is used to find ‘$X_{200}(w)$’.The corresponding plots of ‘$X_{10}(w)$’ and ‘$X_{200}(w)$’ are as shown in *Figure 3.5.2* below. We can observe from the figure below that ‘$X_{200}(w)$’ contains more signal information than the $X_{10}(w)$’.

Calling the function in command window, expand33compare(0.8);

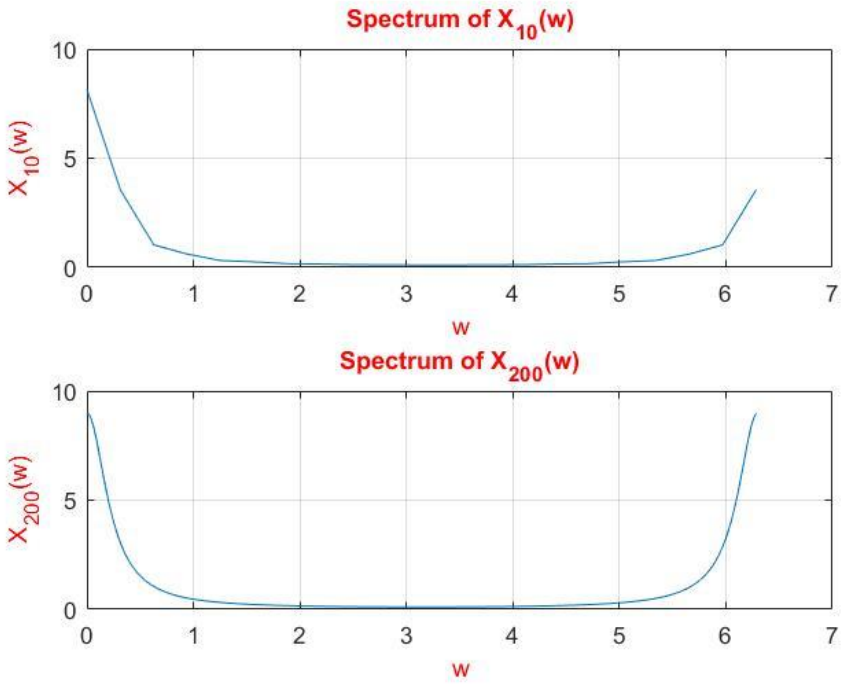

Figure 3.5.2: Comparison of Spectrums $X_{10}(w)$ and $X_{200}(w)$.

Matlab Code:

```
function expand34compare(a)                    % Defining a function with input 'a'
subplot(2,1,1)                                 % Dividing the current plot into subplot
compare_ideal_window(a);                       % Calling function from Exercise 3.4
n = -100:100;                                  % Range of the integer interval
w = linspace(0, 2*pi, length(n));              % Dividing into parts of length n between 0-2*pi
n_200 = -200:1:200;                            % Range of the integer interval
w_200= linspace(0,2*pi,length(n_200));% Dividing into parts of length n_200 between 0-2*pi
X_w = (1 - a^2)./(1- 2*a*cos(w) + a^2);        % Defining the spectrum
subplot(2,1,2)                                 % Dividing the current plot into subplot
plot(w,X_w)                                    % Plot corresponding  spectrum
hold on                                        % Holding the plot for further changes
grid on                                        % To add grid lines in the plot
x_n_200 = a.^(abs(n_200));                     % Defining the real valued series
X_w_200= abs(fft(x_n_200));                    % Applying 'fft' to the x_n_200
plot(w_200,X_w_200);                           % Plot corresponding spectrum
grid on                                        % To add grid lines in the plot
xlabel('Normalised frequency','color','r');    % Labelling the x-axis
ylabel('Magnitude','color','r');               % Labelling the y-axis
title('Ideal spectrum vs Spectrum of windowed series','color','r');    % Giving title to the plot
legend('X(w)','X_{200}(w)')                    % Adding legend to the plot
hold off                                       % Turning OFF hold on function
end                                            % Ends the function
```

Note: The function 'fft' gives the complex values, so to avoid the warning with the MATLAB code stating the Imaginary values are being omitted, so the function 'abs' was used to function the absolute value of the complex value, calculating the magnitude of the real and imaginary part, proper plotting is done.

In the above code, function 'expand34compare(a)' has a single input variable 'a' and it is expected to be in the range (-1,1) i.e., $-1 < a < 1$. we called compare_ideal_window(a) from Exercise 3.4.

At first ideal spectrum 'X(w)' which is the fourier transform of 'x(n)' is used for the same the interval range of 'X(w)' is defined as [-100,100].Later, spectrum of windowed series '$X_{10}(w)$' and '$X_{200}(w)$' are found which are the fourier transforms of '$x_{10}(n)$' and '$x_{200}(n)$'. Whose interval ranges are [-10,10] and [-200,200] respectively are found. Comparison study is as follows:

Comparison of X(w) and $X_{10}(w)$:

Here, we are losing some signal data as explained in Exercise 3.4

Comparison of X(w) and $X_{200}(w)$:

Here ideal spectrum X(w) is defined in interval [-100,100] and windowed signal $X_{200}(w)$ between [-200,200]. Sampling theorem condition is met as explained in Exercise 3.4. So, we are not losing any signal data.

Comparison of $X_{10}(w)$ and $X_{200}(w)$:

The $X_{200}(w)$ contains more signal data without any loss than $X_{10}(w)$ as explained in Exercise 3.4.

Calling the function in command window, expand34compare(0.8);

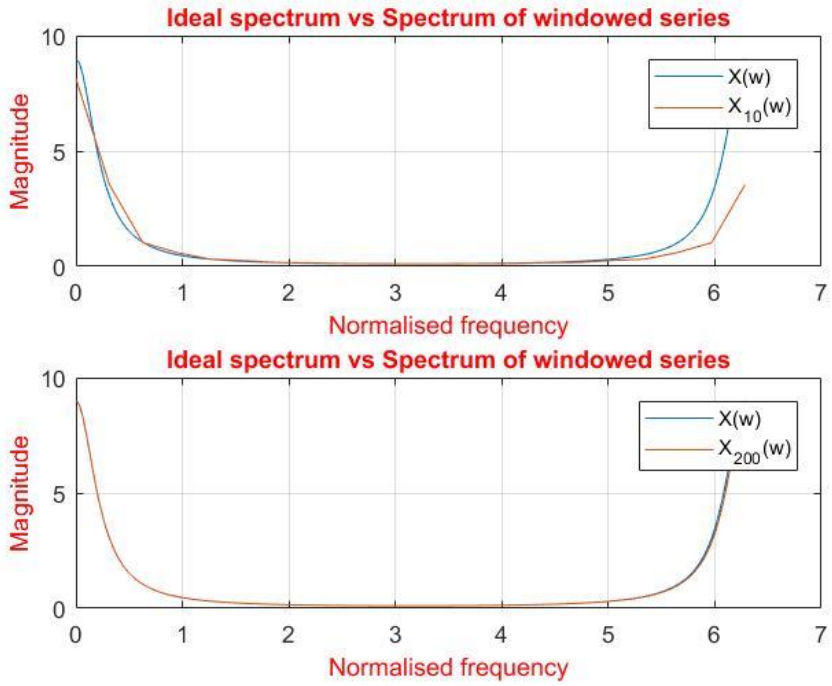

Figure 3.5.3: Comparison of spectrums X(w), $X_{10}(w)$ and $X_{200}(w)$.

4 Optional Exercises

Optional Exercise 4.1: Sampling the spectrum

Sample spectrum $X(\omega)$applying sampling periods $N_b = 21$ and $N_c = 101$ and visualize the results $X_{N_b}(\omega)$and $X_{N_c}(\omega)$. Before sampling multiply the spectrum $X(\omega)$with $e^{j\omega(N_b-1)}$respectively $e^{j\omega(N_c-1)}$.

Matlab Code:

```
function samplingthespectrum(a)          % Defining the function name with input 'a'
N_b = 21;                                % Defining sampling period
N_c = 101;                               % Defining sampling period
w1 = linspace(0, 2 * pi, 21);            % Defining spectrum range and dividing into N_b parts
w2 = linspace(0, 2 * pi, 101);           % Defining spectrum range and dividing into N_c parts
X_w_1 = (1 - a^2)./(1- 2*a*cos(w1) + a^2);     % Defining spectrum
X_w_2 = (1 - a^2)./(1- 2*a*cos(w2) + a^2);     % Defining spectrum
X_w_new1 = X_w_1 .* (exp((1i*w1)*(N_b-1))); % Multiply the spectrum X(ω)with e^jω(N_b−1)
X_w_new2 = X_w_2 .* (exp((1i*w2)*(N_c-1))); % Multiply the spectrum X(ω)with e^jω(N_c−1)
figure;                                  % Opening the figure window
hold on;                                 % Turning ON hold on function
subplot(2,1,1);                          % Dividing the current plot into subplot
plot(w1, abs(X_w_new1));                 % Plot corresponding magnitude spectrum
title('Magnitude plot of X(w) for sampling period N_{b}=21');     % Giving title to the plot
xlabel('Normalised frequency');          % Labelling the x-axis
ylabel('|X_{N_{b}}(w)|');                % Labelling the y-axis
grid on;                                 % To add grid lines in the plot
subplot(2,1,2);                          % Dividing the current plot into subplot
plot(w1, angle(X_w_new1));               % Plot corresponding phase spectrum
title('Phase plot of X(w) for sampling period N_{b} = 21');       % Giving title to the plot
xlabel('Normalised frequency');          % Labelling the x-axis
ylabel('<X_{N_{b}}(w)');                 % Labelling the y-axis
grid on;                                 % To add grid lines in the plot
hold off;                                % Turning OFF hold on function
figure;                                  % Opening the figure window
hold on;                                 % Turning ON hold on function
subplot(2,1,1);                          % Dividing the current plot into subplot
plot(w2, abs(X_w_new2));                 % Plot corresponding magnitude spectrum
title('Magnitude plot of X(w) for sampling period N_{c}=101');    % Giving title to the plot
xlabel('Normalised frequency');          % Labelling the x-axis
ylabel('|X_{N_{c}}(w)|');                % Labelling the y-axis
grid on;                                 % To add grid lines in the plot
subplot(2,1,2);                          % Dividing the current plot into subplot
plot(w2, angle(X_w_new2));               % Plot corresponding phase spectrum
title('Phase plot of X(w) for sampling period N_{c} = 101');      % Giving title to the plot
xlabel('Normalised frequency');          % Labelling the x-axis
ylabel('<X_{N_{c}}(w)');                 % Labelling the y-axis
grid on;                                 % To add grid lines in the plot
```

```
hold off;                          % Turning OFF hold on function
end                                % Ends the function
```

Initially spectrum X(w) is multiplied with $e^{j\omega(N_b-1)}$ and $e^{j\omega(N_c-1)}$ respectively. Then sampled with sampling periods $N_b = 21$ and $N_c = 101$. For sampling period $N_b = 21$ some part of signal data is found to be missing because sampling theorem condition is not met and for $N_c = 101$ includes more signal information is observed. The magnitude and phase plots for the spectrum X(w) with sampling periods $N_b = 21$ and $N_c = 101$ are plotted as shown in Figure 4.1.1 and Figure 4.1.2 respectively.

Calling the function in command window, samplingthespectrum(0.8);

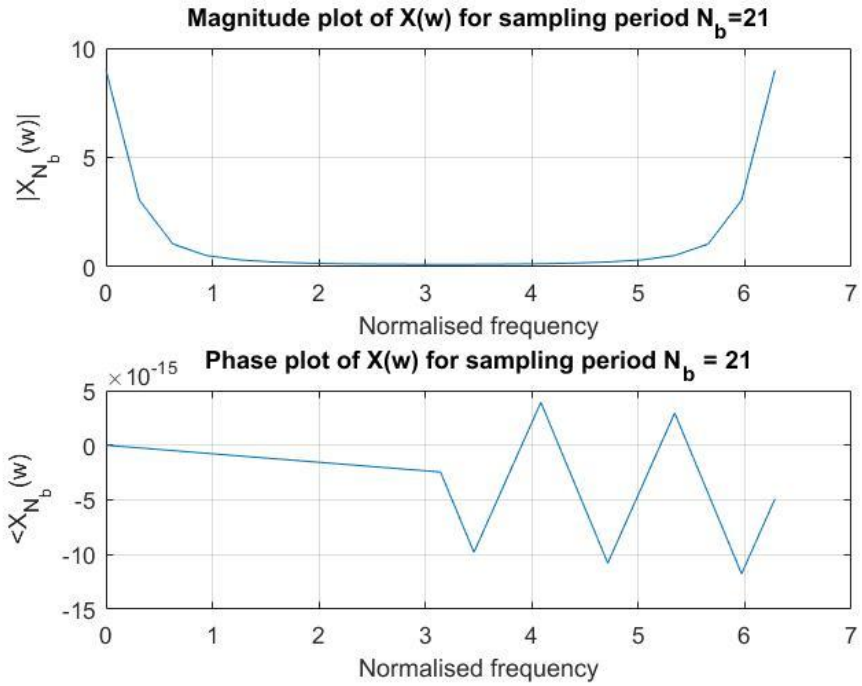

Figure 4.1.1: Magnitude and Phase plots for N_b=21.

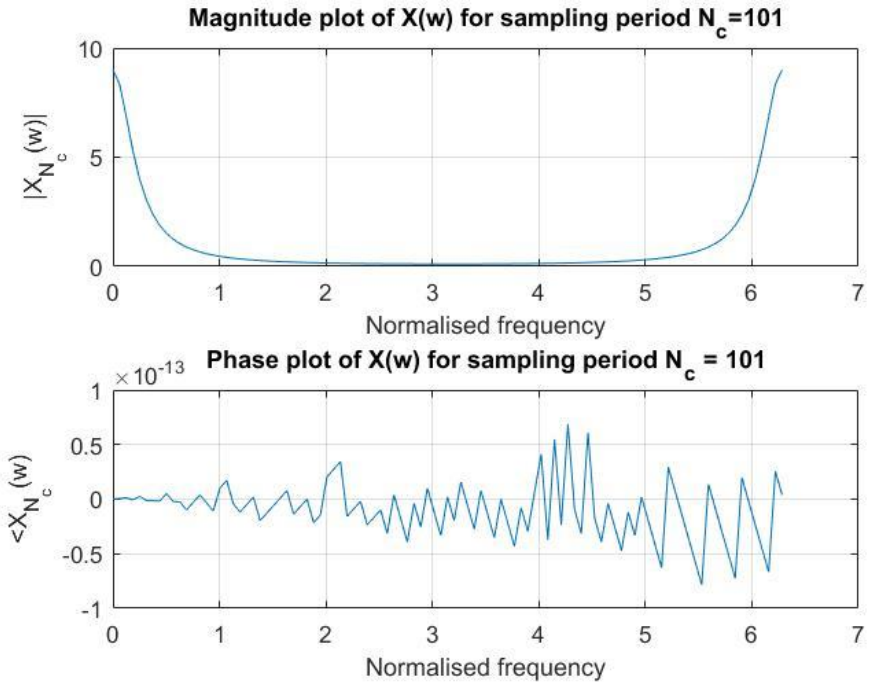

Figure 4.1.2: Magnitude and Phase plots for N_c=101..

Optional Exercise 4.2: Reconstruction of the sampled spectrum

Apply an inverse Fourier transformation on the sampled spectra from the last task (Matlab : ifft). Re-transform these time signals back into the frequency domain (Matlab: fft) and visualize the spectra in the interval [0,2pi]. For ft use "zero padding" and a length of Nk=10499.

Matlab Code:

```
function reconstruction_spectrum(a)          % Defining the function name with input 'a'
N_b = 21;                                    % Defining sampling period
N_c = 101;                                   % Defining sampling period
w1 = linspace(0, 2 * pi, 21);                % Defining spectrum range and dividing into N_b parts
w2 = linspace(0, 2 * pi, 101);               % Defining spectrum range and dividing into N_c parts
X_w_1 = (1 - a^2)./(1- 2*a*cos(w1) + a^2);   % Defining spectrum
X_w_2 = (1 - a^2)./(1- 2*a*cos(w2) + a^2);   % Defining spectrum
X_w_new1 = X_w_1 .* (exp((1i*w1)*(N_b-1)));  % Multiply the spectrum X(ω)with e^{jω(N_b-1)}
X_w_new2 = X_w_2 .* (exp((1i*w2)*(N_c-1)));  % Multiply the spectrum X(ω)with e^{jω(N_c-1)}
x_n_21 = ifft(X_w_new1);                     % Applying Inverse FT
x_n_101 = ifft(X_w_new2);                    % Applying Inverse FT
N_k = 10499;                                 % Defining sampling period for zero-padding
w3 = linspace(0, 2*pi, N_k);                 % Defining spectrum range and dividing into N_k parts
X_w_retransform1 = fft(x_n_21, N_k);         % Applying DFT using zero padding
X_w_retransform2 = fft(x_n_101, N_k);        % Applying DFT using zero padding
figure;                                      % Opening the figure window
hold on                                      % Turning ON  hold on function
plot(w3, abs(X_w_retransform1));             % Plot corresponding spectrum
title('Reconstruction of the sampled spectrum X_{N_b}(w)');   % Giving title to the plot
xlabel('Normalized frequency');              % Labelling the x-axis
ylabel('Magnitude');                         % Labelling the y-axis
figure;                                      % Opening the figure window
plot(w3, abs(X_w_retransform2));             % Plot corresponding spectrum
title('Reconstruction of the sampled spectrum X_{N_c}(w)');   % Giving title to the plot
xlabel('Normalized frequency');              % Labelling the x-axis
ylabel('Magnitude');                         % Labelling the y-axis
end                                          % Ends the function
```

In the above Matlab code, the function 'ifft' is used and the inverse fourier transform of the resultant signal from the Exercise 4.1 is found.The spectrums are then retransformed and visualized for different values are shown in *Figure 4.2.1* and *Figure 4.2.2* below.

we reconstructed the sampled spectrum by using zero-padding. By observing *Figure 4.2.1* and *Figure 4.2.2* below, we can say that for high sampled values of a signal, zero padding provides more signal information in amplitude and vice versa.

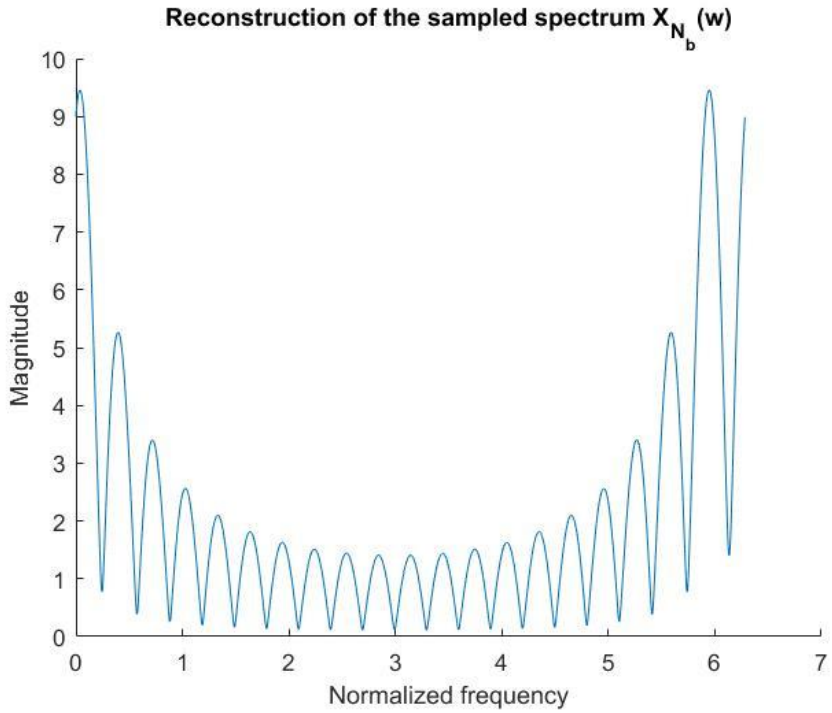

Figure 4.2.1: Reconstruction of the sampled spectrum $X_{Nb}(w)$.

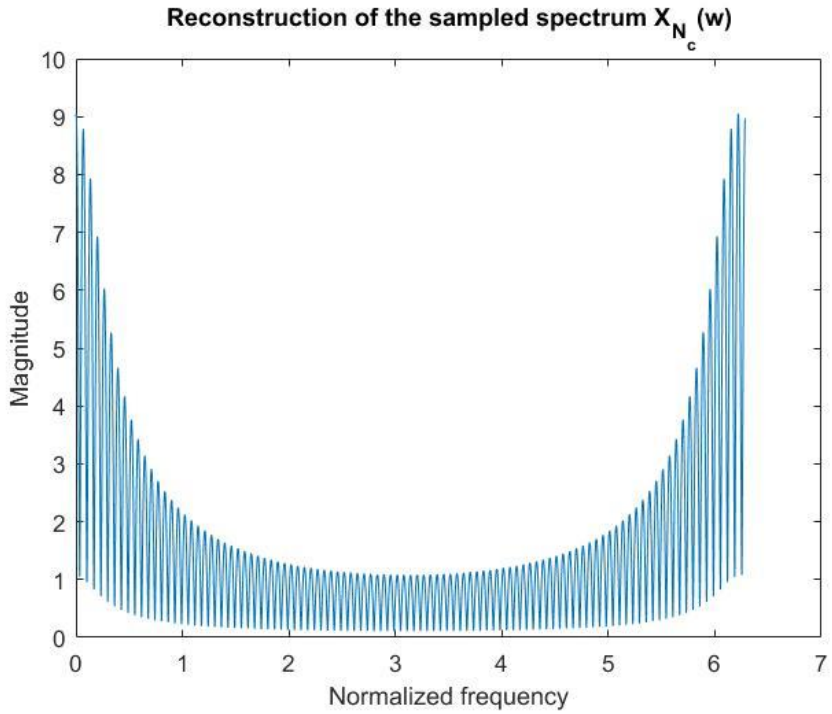

Figure 4.2.2: Reconstruction of the sampled spectrum $X_{Nc}(w)$.

5 References

[1] z-Transform http://ee.eng.usm.my/eeacad/shahrel/dsp/EEE443Lect7.pdf

[2] What is Matlab – Where and Why to Use IT http://www.matlabtips.com/what-is-matlab-where-how-and-when-to-use-it/

[3] Sampling theorem https://de.mathworks.com/matlabcentral/mlc-downloads/downloads/submission/47700/versions/2/screenshot.jpg