18.03.2025

Krzysztof Smykla

Lab 1 Report

Problem 1 – Unit Impulse Function δ*[n]:*

The problem consisted of graphing the discrete-time unit impulse function and the unit step function u[n] using built-in MATLAB functions *ones()* and *zeros().* The specified domain for both functions was **n ∈ [-10,10]**.

1b:

Modifying the code provided in the description I generated the function **δ[n - 2]** using the built-in MATLAB function *circleshift()* which circularly shifts the elements in array A by K positions.

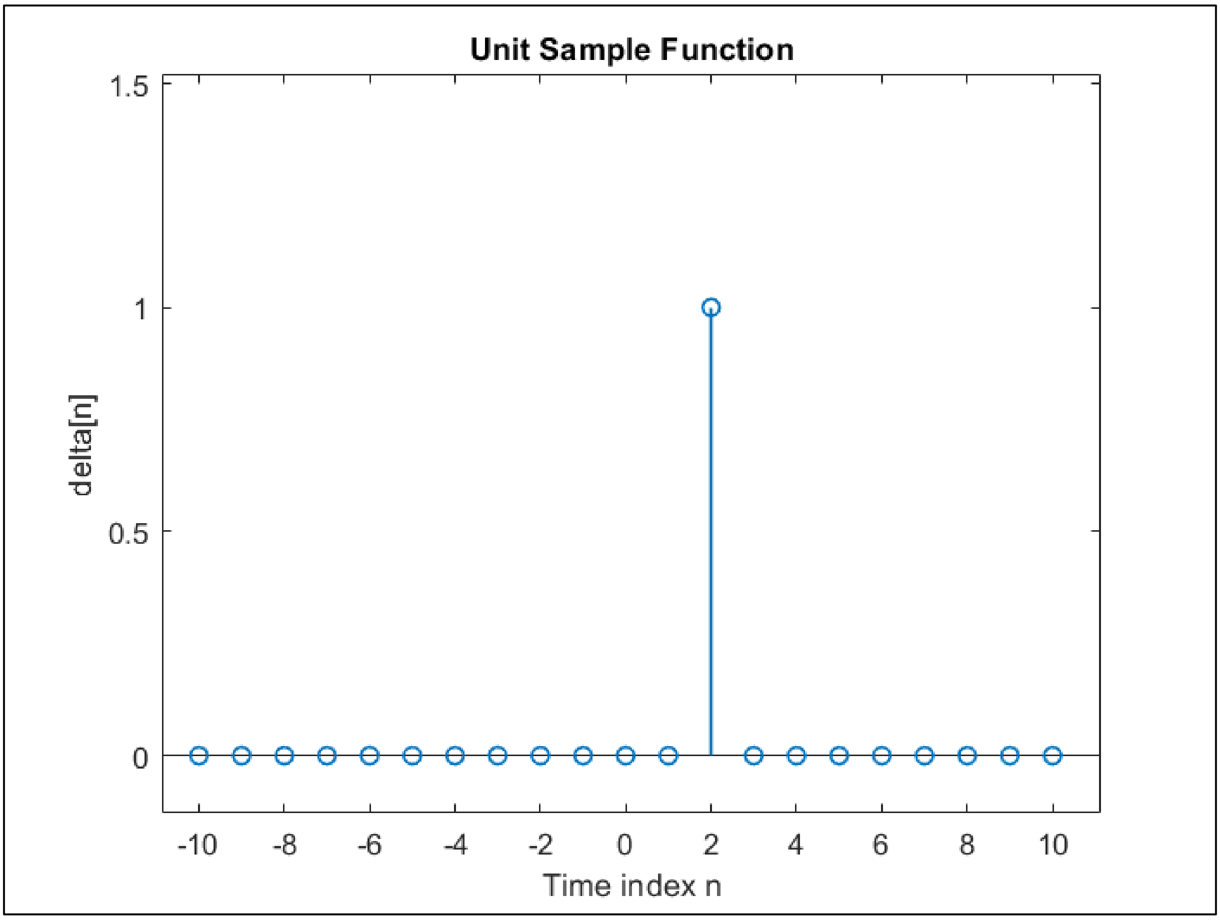
(MATLAB documentation: https://uk.mathworks.com/help/releases/R2024b/MATLAB/ref/circshift.html)

See MATLAB code below:

```
 % Exercize 1

% P1a

% generate the signal delta[n] and plot it

clc,clearvars

n = -10: 10;  %values of time domain

delta_n = [zeros(1,10), 1, zeros(1,10)];



%b

figure(1)

stem(n, circshift(delta_n, 2), LineWidth=1); % delta impulse shifted circularly by 2

axis([-10, 10, 0, 1.5]);

title('Unit Sample Function');

xlabel('Time index n');

ylabel('delta[n]');
```

The code provided generates the modified unit impulse function and shifts it by a vector *v*[2,0].

Fig. 1:



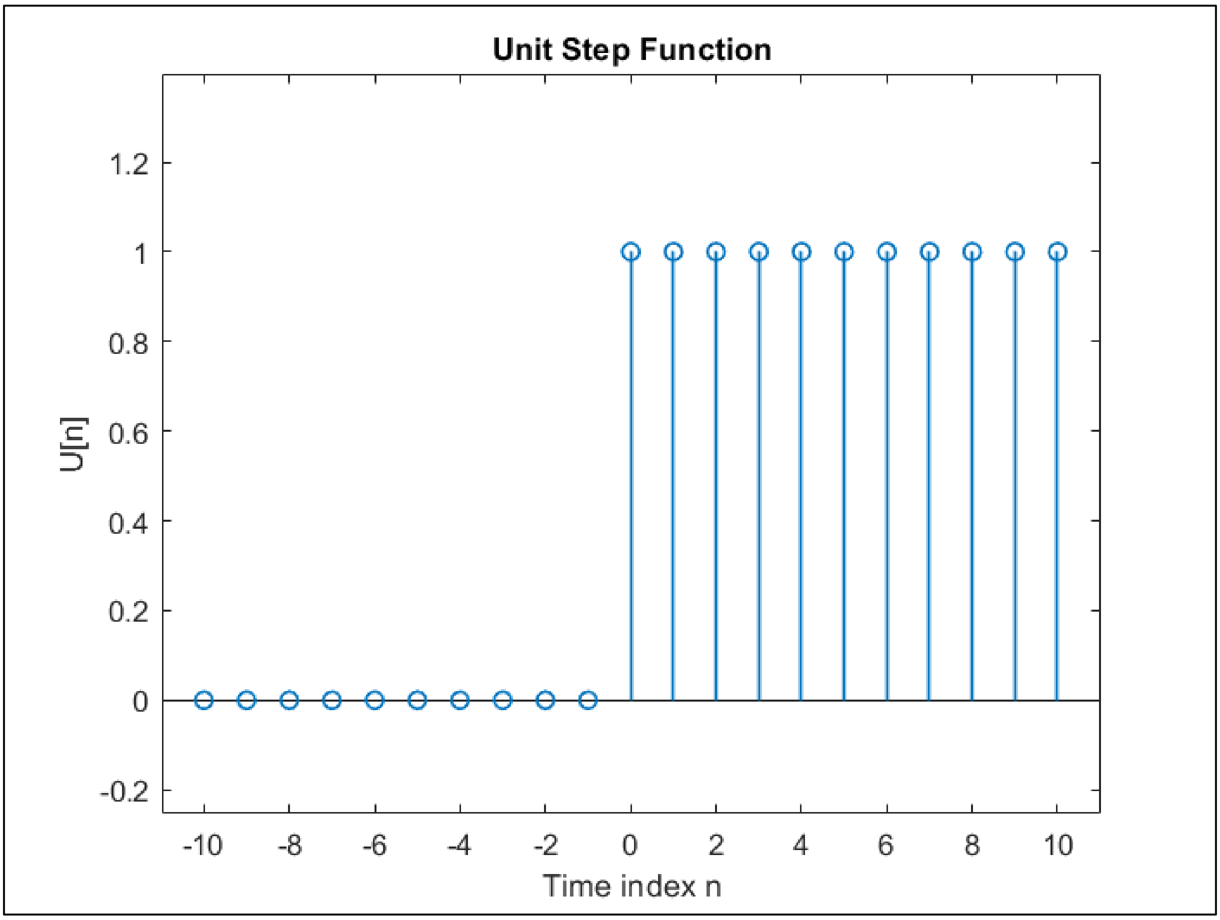
1c:

In point c the task was to create the unit step function u[n] also using the ones() and zeros() functions as before on the range **n ∈ [-10,10]**.

```
%c

figure(2)

u_step = [zeros(1,10), 1, ones(1,10)];

stem(n,u_step,LineWidth=1); % unit step function plot

axis([-10, 10, 0, 1.5]);



title('Unit Step Function');



xlabel('Time index n');

ylabel('U[n]');
```

The code generates the function and plots it on a separate figure.

Fig. 2:



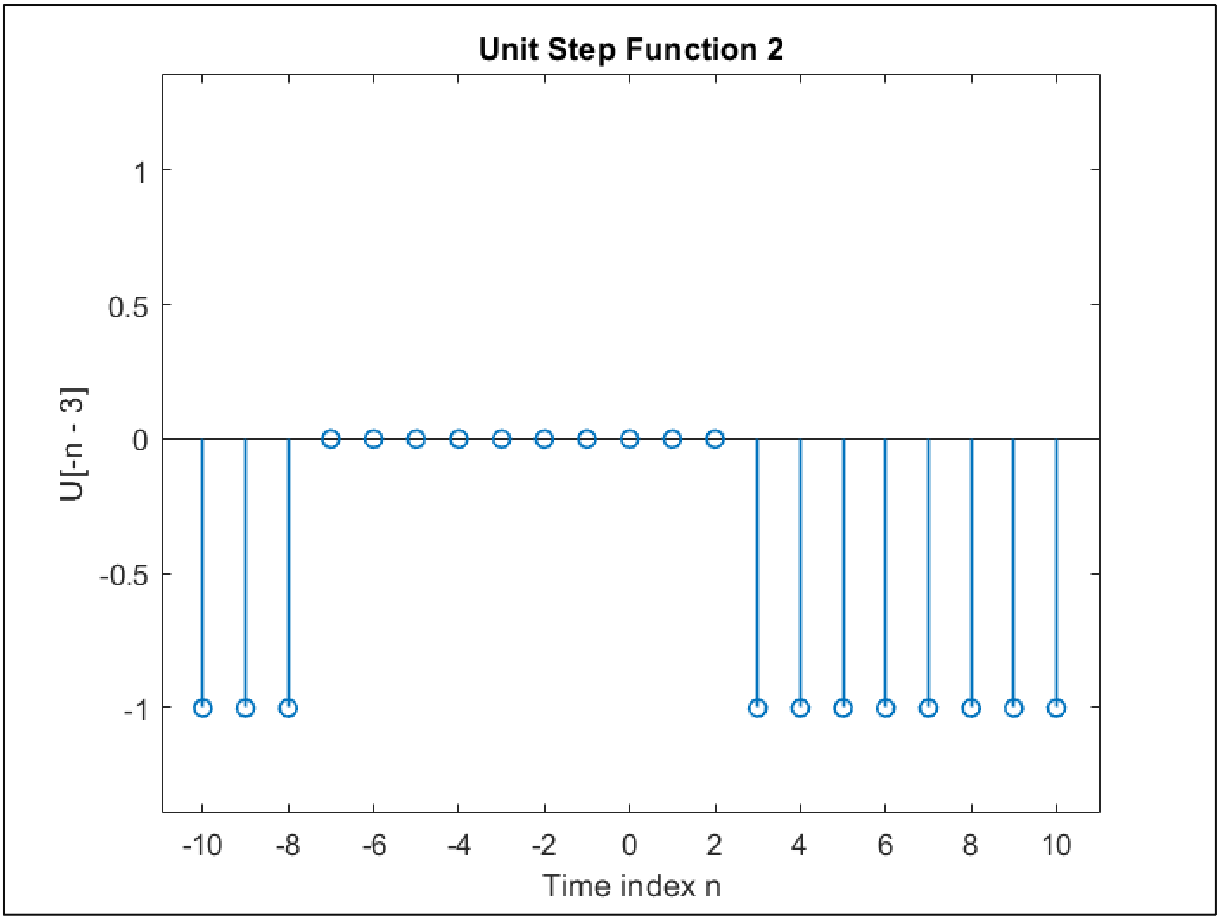
1d:

In point d I was asked to generate a plot for the function U[-n-3].

MATLAB code is provided below:

```
%d

figure(3)

stem(n, circshift(-u_step, 3), LineWidth=1); % unit step shifted by v[3, -1]

axis([-10, 10, 0, 1.5]);


title('Unit Step Function 2');


xlabel('Time index n');

ylabel('U[-n - 3]');
```

Fig. 3:


Unit Step Function 2
U[-n - 3]
1
0.5
0
-0.5
-1
-10
-8
-6
-4
-2
0
2
4
6
8
10
Time index n

Problem 2 – Cosine signal (discrete-time):

Problem 2 asks to generate a discrete-time cosine signal, plot said signal and analyze the signal's fundamental period. The problem also asks to describe the use of the *grid()* function in MATLAB.

2a:

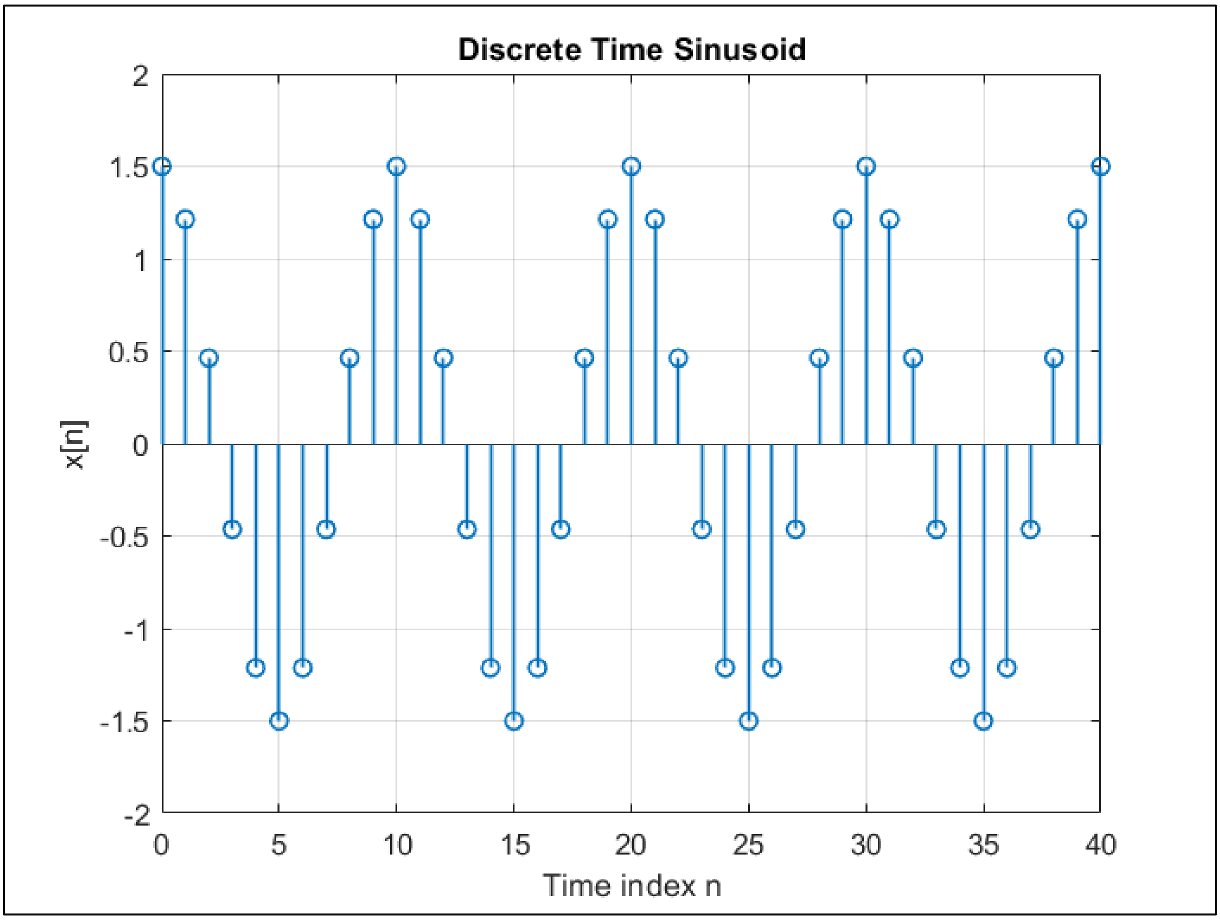
Generate and plot a discrete-time cosine signal.

The code below is the solution for part 'a' of the problem as provided in the problem description.

```
 % P2a



% generate and plot a discrete-time cosine signal

clc, clearvars



n = 0:40;  % values of the time variable

w = 0.1*2*pi;  % frequency of the sinusiod.

phi = 0; % phase offset.

A = 1.5; % amplitude

xn = A * cos(w*n - phi); % signal formula



figure(1)

grid("on") % using the grid function

stem(n, xn, LineWidth=1);  % sinusoid plot in discrete [n] domain

axis([0, 40, -2, 2]);

grid;

title('Discrete Time Sinusoid');

xlabel('Time index n');

ylabel('x[n]');
```

The code provided generates the following figure:

Fig. 4:



2b:

Part ‘b’ asks to find the length of the signal plotted above. To do that I used the *length()* function built into MATLAB.

```
%b

l = length(xn) % signal length



fprintf('The length of the signal is %d \n', l )
```

Output:

l =

41

The length of the signal is 41

2c:

To solve this part of the problem, I used the built-in *max()* function to find the maximum value of the signal defied in the variable *xn*. The variable called *indices* stores all the occurrences of the maximum value in a vector with the help of the *find()* function. I later find the sum of all the samples present between two maxima `sum(abs(indices(1) - indices(2)))`. This allows me to find the number of samples which make up the fundamental period of the discrete-time signal *xn*.

Alternatively, one can use the “Signal Processing toolbox” addon to MATLAB, which includes the functionality for signal processing. I used the function *findpeaks()* which finds the local maxima (peaks) of the signal vector.

(MATLAB documentation: https://uk.mathworks.com/help/releases/R2024b/signal/ref/findpeaks.html). The function *findpeaks()* was introduced in Signal Processing Toolbox in R2007b

MATLAB code below:

```
 %c

% The fundamental period can be calculated by finding the difference between the local maxima of the signal.

% Later count the number of samples from crest to crest



maxYValue = max(xn);

indices = find(xn == maxYValue); % Get all indices where max value occurs

if length(indices) > 1

    xValues = sum(abs(indices(1) - indices(2))); % Difference of first two occurrences of the maximum

else

    xValues = indices; % If only one occurrence, return the index itself

end

fprintf("The fundamental period is %d samples \n",xValues)



% Alterantively, simply count the number of samples present in the signal

% from crest to crest.
```

The value of the fundamental period is stored in the *xValues* variable. This number can also be calculated by simply counting the number of individual samples present between two crests.

2d:

The final point asks to describe the purpose of the *grid()* function in MATLAB.

The purpose of the *grid()* function is to display a grid inside a given figure in order to improve visualization.

Problem 3 – Sine signal (discrete-time):

Problem 3 the task is to use MATLAB to generate and plot the discrete-time signal $x[n] = \sin(\omega_0 n)$ for the following values of ω_0:

$-29\pi/8$, $-3\pi/8$, $-\pi/8$, $\pi/8$, $3\pi/8$, $5\pi/8$, $7\pi/8$, $9\pi/8$, $13\pi/8$, $15\pi/8$, $33\pi/8$, and $21\pi/8$

The main challenge I faced when solving the problem was automatically processing the values of ω_0 to generate consecutive sin plots and visualize them aesthetically. I used a for loop to generate a sine function for every value k in the list of values above. The I used the if statement to display the plots in groups of 4 on separate figures.

The corresponding MATLAB code is provided below.

```
 clc, clearvars


% P3a


% Use MATLAB o generate and plot the discrete-time-signal x[n] = sin(wn)

% for the following values of w:


% -29pi/8,  -3pi/8, -pi/8, pi/8, 3pi/8, 5pi/8, 7pi/8, 9pi/8, 13pi/8,

% 15pi/8, 33pi/8, and 21pi/8 .



n = 0:63;  % discrete-time domain

k_values = [-29, -3, -1, 1, 3, 5, 7, 9, 13, 15, 33, 21];

numPlots = length(k_values);

plotsPerFigure = 4;  % We want a 4x1 grid in each figure
```

```
for i = 1:numPlots

    % Open a new figure every time we start a new group of 4 plots

    if mod(i-1, plotsPerFigure) == 0

        figure;

    end



    % Determine subplot index within the current figure (1 to 4)

    subplotIndex = mod(i-1, plotsPerFigure) + 1;

    subplot(plotsPerFigure, 1, subplotIndex);



    % Compute the angular frequency and the corresponding sinusoid

    w = k_values(i) * pi/8;

    x = sin(w * n);



    % Plot the sinusoid using stem

    stem(n, x, 'LineWidth', 1);

    title(sprintf('%d\\pi/8', k_values(i)));

    xlabel('Time index n');

    ylabel('x[n]');

    axis([min(n), max(n), -2, 2]);

    grid on;

end




% After analyzing the visualizations, I concluded that the graph of the

% sinusoid changes its shape every 2π/8 of a rotation.



% The signal repeats (i.e. is periodic) when the argument of the sine

% increases by a multiple of 2π.



% The graphs repeat every 2*k*π rotations where k = 8.
```

The code outputs the following figures:

Fig 5.

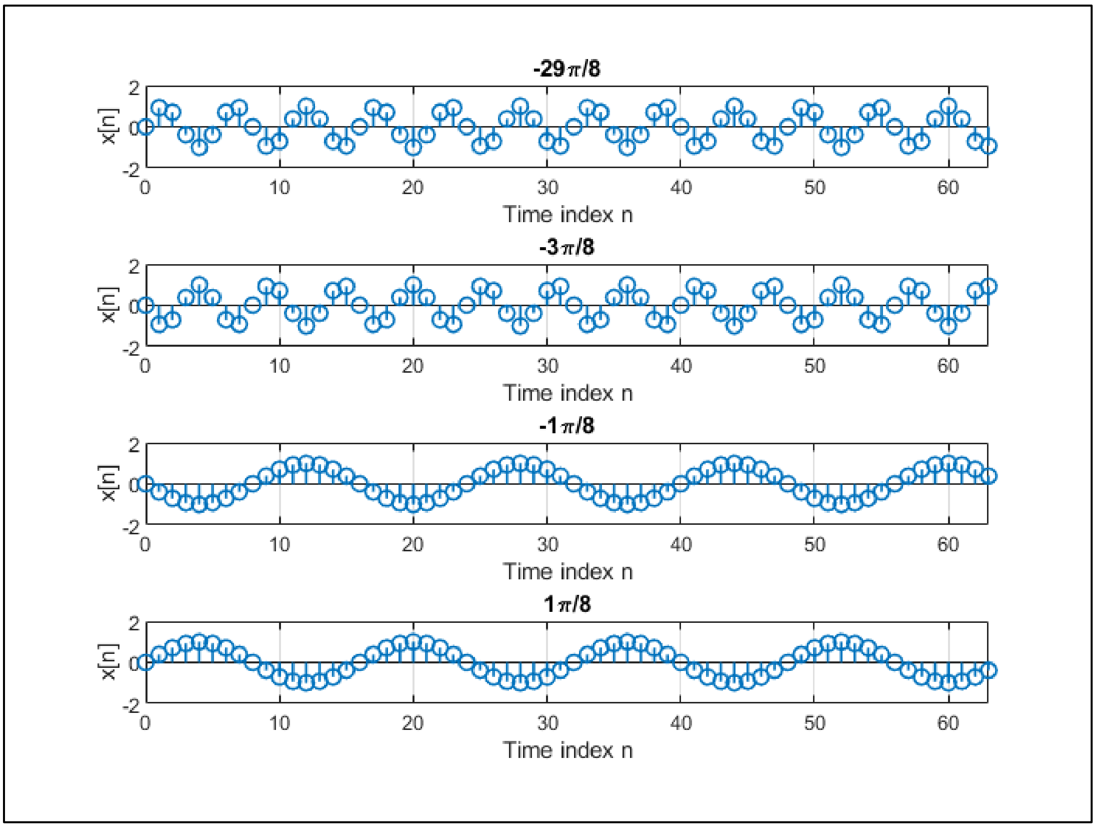


Fig 6.

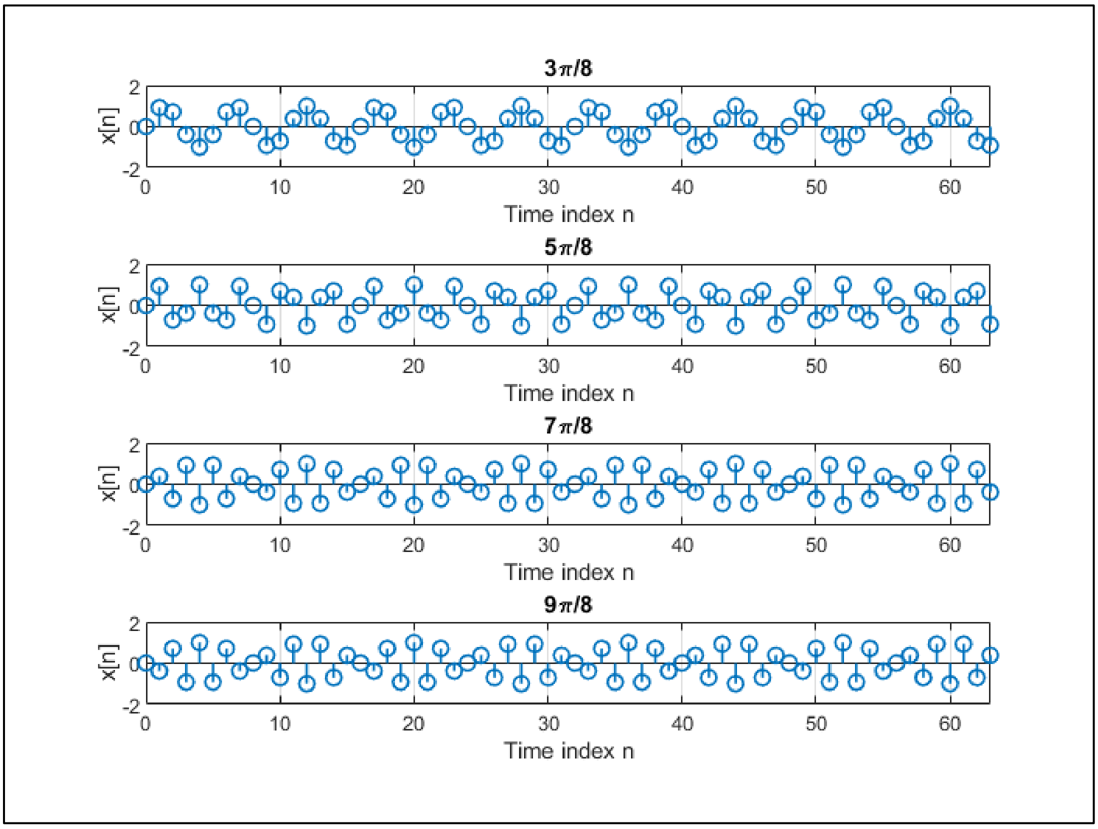
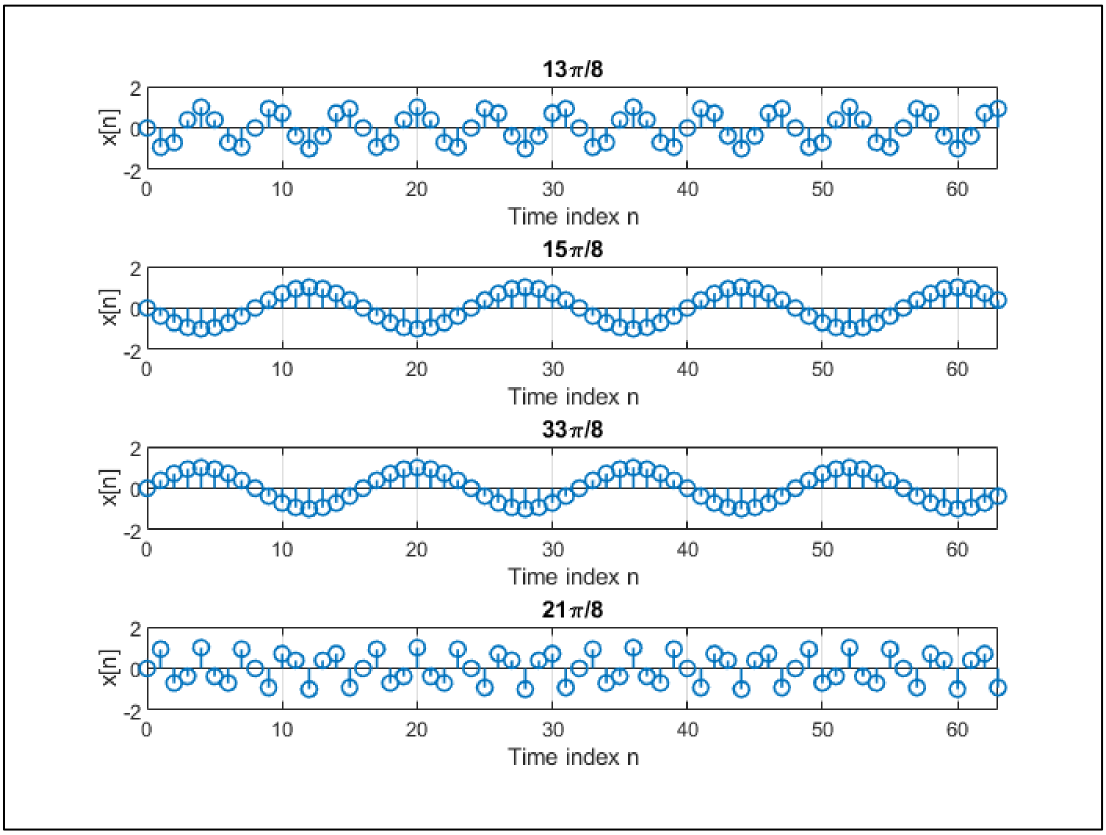


Fig. 7



The questions in problem 3 were:

1. Are any of the graphs from the above part identical to one another?
2. How are the graphs of $x[n] = \sin(\omega_0 n)$ for $\omega_0 = 7\pi/8$ and $\omega_0 = 9\pi/8$ related?

Answers:

Question 1:

The graphs are identical for the following pairs of arguments: (-π/8, 15π/8); (π/8, 33π/8);

(-29π/8, 3π/8) ;(5π/8, 21π/8) ;(-3 π/8, 13 π/8) and (1 π/8, 33 π/8).

Since the fundamental period for a sinusoid is 2*π, in our case the function's shape repeats every 2*k*π/8 where k = 8.

Question 2:

Relationship Between sin(7π/8) and sin(7π/8)

1. Calculate the difference between the two angles:

$9\pi/8 - 7\pi/8 = 2\pi/8 = \pi/4$

This shows that the two angles are separated by π/4 in terms of frequency, but we need to analyze their phase relationship.

2. Using trigonometric identities:
   We express the angles in terms of π radians:

For $\sin(7\pi/8)$:

$$\sin(7\pi/8) = \sin(\pi - \pi/8)$$

Using the identity:

$$\boldsymbol{\sin(\pi - \theta) = \sin(\theta)}$$

$\sin(7\pi/8) = \sin(\pi/8)$

For $\sin(9\pi/8)$:

$$\sin(9\pi/8) = \sin(\pi + \pi/8)$$

Using the identity:

$$\boldsymbol{\sin(\pi + \theta) = -\sin(\theta)}$$

$\sin(9\pi/8) = -\sin(\pi/8)$

Conclusion:

Since,

$\sin(7\pi/8) = \sin(\pi/8)$ and $\sin(9\pi/8) = -\sin(\pi/8)$

it follows that:

$\sin(9\pi/8) = -\sin(7\pi/8)$

This means the two functions are negatives of each other, which corresponds to a 180° phase shift (or a sign flip).

Due to the 180° phase shift the **amplitude** has flipped sign, but the **magnitude** remains the same.

Problem 4 – Sine signal (discrete-time):

In the final problem, the code provided in the description generated the following graphs:

Fig. 8

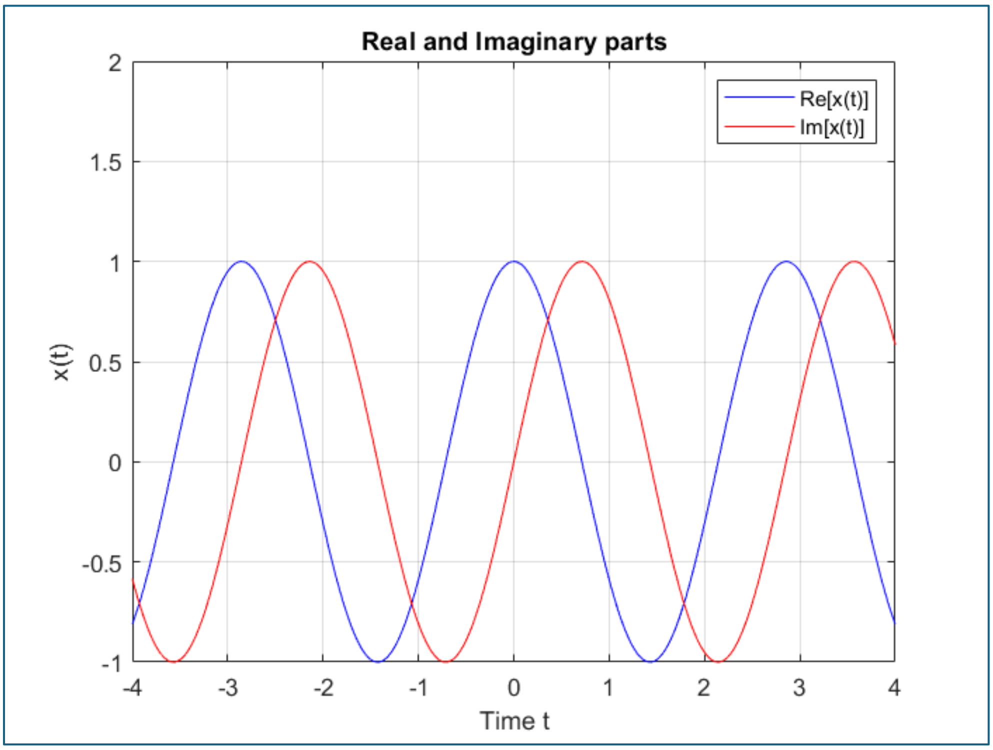
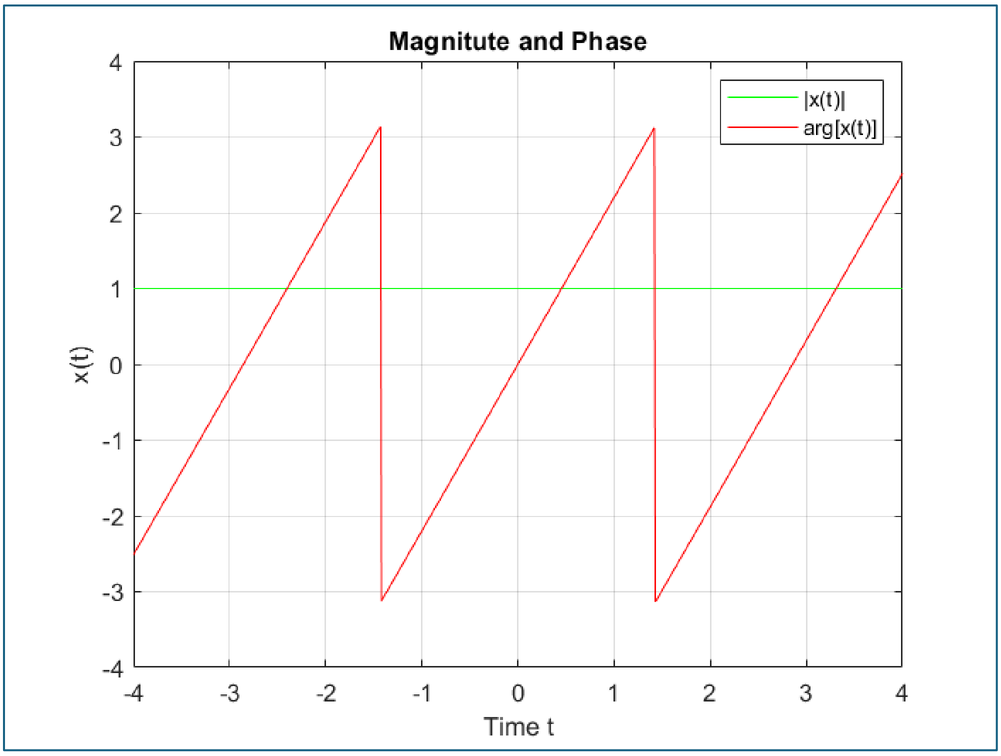


Fig. 9



The second part of the problem asks us to use similar MATLAB statements to generate the continuous-time damped exponential signal:

$$x(t) = 3e^{-t/2}e^{j8t}$$

for 0 ≥ t ≥ 4.

and to plot its magnitude and phase.

The code I used to solve this problem is provided below:

```
 % % P4a cont.

clc, clearvars

% New signal xt_2



t_2 = 0:0.01:4; % new time domain

w2 = 8; % new fequency

xt_2 = 3 .* exp(- t_2 ./2 ) .* exp(1i*w2*t_2)  % new complex signal



xt_2R = real(xt_2);

xt_2I = imag(xt_2);



figure(3); % Open new figure



plot(t_2, xt_2R, '-b'); % '-b' means 'solid blue line'



 axis([-4, 4, -1, 2]);

 grid on;

 hold on;   % add more curves to the same graph

 plot(t_2, xt_2I, '-r');  % solid red line

 title('Real and Imaginary parts');

 xlabel('Time t_2');

 ylabel('x(t_2)');

 legend('Re[x(t_2)]', 'Im[x(t_2)]');

 hold off;
```

```
 mag2 = abs(xt_2);

 phase2 = angle(xt_2);



figure(4);

plot(t_2, mag2, '-g'); % solid green line

 grid on;

 hold on;

 plot(t_2, phase2, '-r'); % solid red line

 title('Magnitute and Phase');

 legend('|x(t_2)|', 'arg[x(t_2)]');

 xlabel('Time t_2');

 ylabel('x(t_2)');

 hold off;
```

The code defines the new complex signal in the variable `xt_2` and generates the graphs of the signal as well as the signal's phase and magnitude.

The graphics are shown below:

$X(t_2) = 3e^{-t/2}e^{j8t}$

Fig. 10

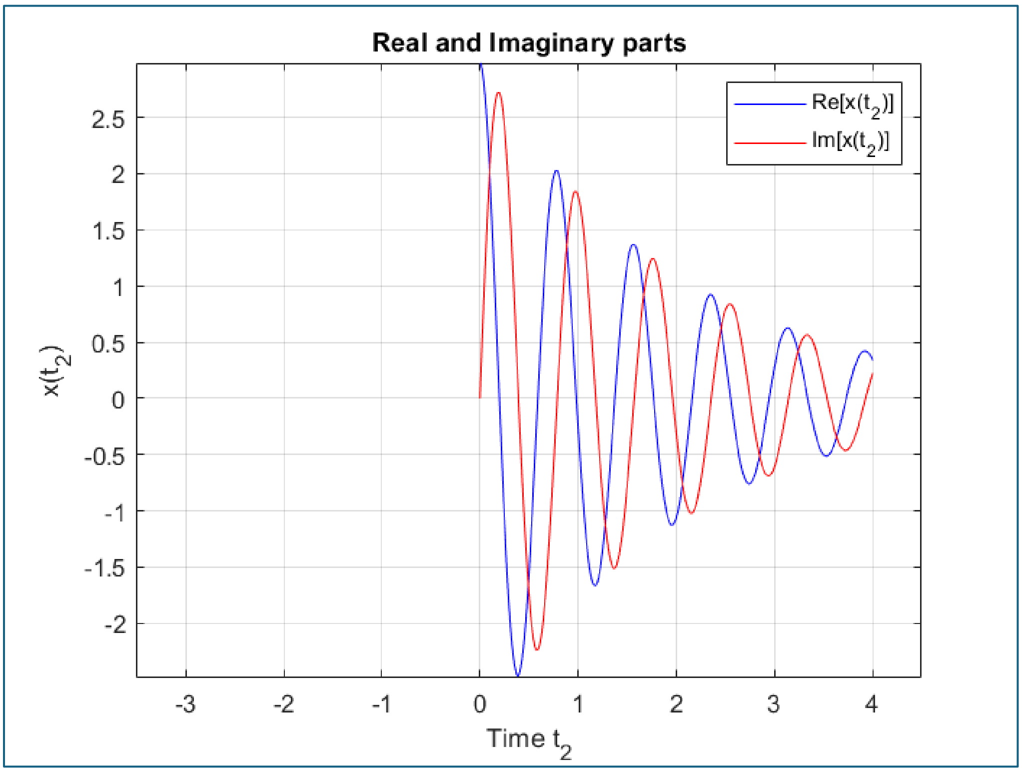
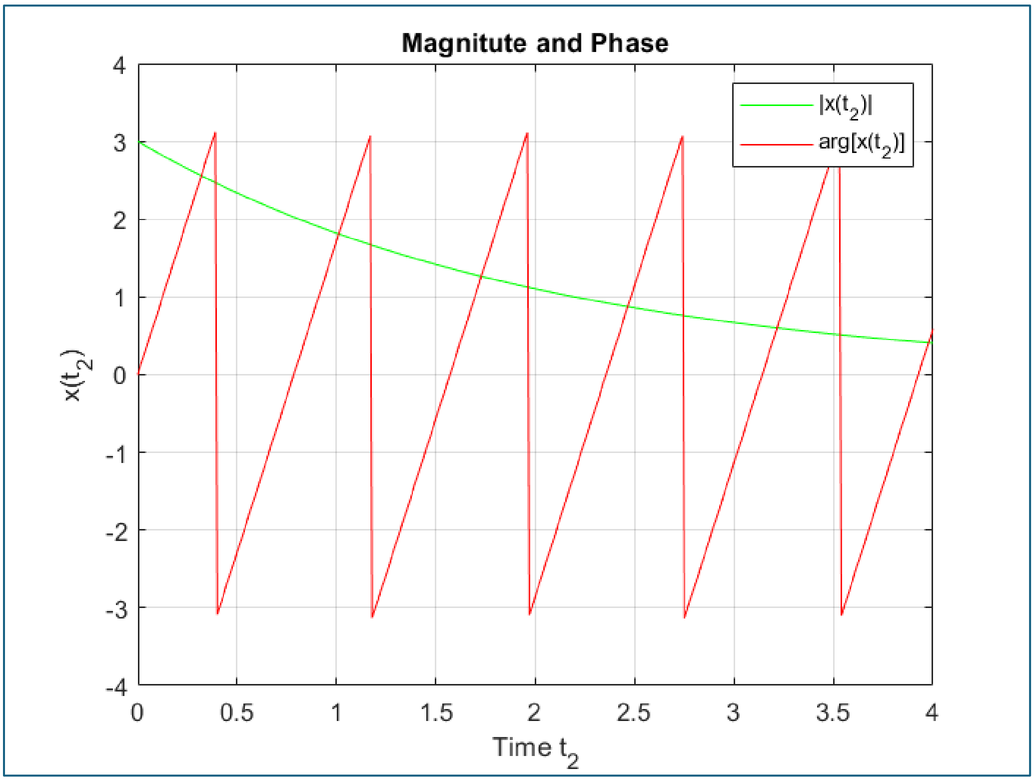


Fig. 11



This concludes the final problem in the assignment.

Conclusion:

When working out the solutions to the problems given in the assignment, I used MATLAB's documentation extensively to look up several built-in functions provided in the base version of MATLAB (v. 2024b). I decided to use the MATLAB "Signal Processing toolbox" addon in problem to simplify the code and improve readability. The problem I struggled most with was problem 3 where I needed to come up with a way to generate the corresponding sinusoidal plots automatically for all the arguments provided and correctly explain my solution.

Krzysztof Smykla (album nr.: 417410)