电子科技大学格拉斯哥学院

Glasgow College UESTC

标 准 实 验 报 告

Lab Report

（实验）课程名称：信号与系统

(LAB) Course Name：Signals and Systems

电子科技大学教务处制表

Lab Report 5

Student Name：Changgang Zheng 郑长刚
UESTC ID：2016200302027
UoG ID：2289258Z
Instructor：Huijuan Wu 吴慧娟
Location：UESTC, ChengDu, SiChuan Province, China
Date： May 7, 2018

1. **Laboratory name：**Signals and Systems
2. **Project name：**Represent signals using MATLAB
3. **Lab hours：**2
4. **Theoretical background：**

   1. The basic concepts of signals and systems arise in a variety of contexts, from engineering design to financial analysis. In this lab1, you will learn how to represent, manipulate, and analyze basic signals and systems in MATLAB.
   2. Some basic MATLAB commands for representing signals include: zeros, ones, cos, sin, exp, real, imag, abs, angle, linspace, plot, stem, subplot, xlabel, ylabel, title.
   3. Some useful commands in Symbolic Math Toolbox are as: zeros, ones, cos, sin, exp, real, imag, abs, anglelinspace, plot, stem, subplot, xlabel, ylabel, title. Symbolic Math Toolbox: sym, subs, ezplot, Conv, filter, lsim.
5. **Objective：**

   Compute the output of causal LTI system characterized by linear constant-coefficient differential/ difference equations.
6. **Description：**

   The following exercises are from the book “John R.Buck, Michael M. Daniel, Andrew C. Singer. Computer Exploration in Signals and Systems —— Using MATLAB.”
   1. Compute the Discrete-time frequency response with freqz. 3.2
   2. Eigen functions of Discrete-Time LTI Systems. 3.4(a)(b)(c)
   3. Synthesizing Signals with the Discrete-Time Fourier Series. 3.5(d)(e)(f)(h)
7. **Required instruments：**

   Computer, MATLAB.
8. **Procedures, Analysis of Lab data & result and Conclusion：**

MATLAB codes and results for the exercises:

■ 3.2 Tutorial: `freqz`

The signals $e^{j\omega n}$ are eigenfunctions of LTI systems. For each value of ω the frequency response $H(e^{j\omega})$ is the eigenvalue of the LTI system for the eigenfunction $e^{j\omega n}$; when the input sequence is $x[n] = e^{j\omega_0 n}$, the output sequence is $y[n] = H(e^{j\omega_0})e^{j\omega_0 n}$. For a causal LTI system described by a difference equation, the command `[H omega]=freqz(b,a,N)` computes the frequency response $H(e^{j\omega})$ at N evenly spaced frequencies between 0 and π, i.e., $\omega_k = (\pi/N)k$ for $0 \le k \le N-1$. The coefficient vectors `a` and `b` specify the difference equation using the same format as Eq. (2.8) in the `filter` tutorial. For the command above, `freqz` returns $H(e^{j\omega_k})$ in `H` and the frequencies ω_k in `omega`. Including the `'whole'` option as `[H omega]=freqz(b,a,N,'whole')` computes the samples of the frequency response $H(e^{j\omega})$ at N evenly spaced frequencies from 0 to 2π, $\omega_k = (2\pi/N)k$ for $0 \le k \le N-1$.

(a). Define `a1` and `b1` to describe the causal LTI system specified by the difference equation $y[n] - 0.8y[n-1] = 2x[n] - x[n-2]$.

(b). Use `freqz` with the coefficients from Part (a) to define `H1` to be the value of the frequency response at 4 evenly spaced frequencies between 0 and π and `omega1` to be those frequencies. The following sample output shows the values each vector should have if you have defined things correctly:

```
>> H1.'
ans =
   5.0000    2.8200 - 1.3705i    1.8293 - 1.4634i    0.9258 - 0.9732i
>> omega1.'
ans =
         0    0.7854    1.5708    2.3562
```

(c). Use `freqz` to define `H2` to be the value of the frequency response at 4 evenly spaced frequencies between 0 and 2π and `omega2` to be those frequencies. The following sample output shows the values each vector should have if you have defined things correctly:

```
>> H2.'
ans =
   5.0000    1.8293 - 1.4634i    0.5556    1.8293 + 1.4634i
>> omega2.'
ans =
         0    1.5708    3.1416    4.7124
```

The code:

```
%Name: Matlab: Signals and Systems Lab 5th
%Author: Changgang Zheng
%Student Number UESTC:2016200302027
%Student Number UoG:2289258z
%Institution: Glasgow College UESCT
%Question: Perform convolution. 3.2

function  problem_1st
    %% problem a
    a1=[1 -0.8];
```

```
b1=[2 0 -1];
%%
%% problem b
N=4;
[H1,omega1]=freqz(b1,a1,N);
H1.'
omega1.'
%%
%% problem c
N=4;
[H2,omega2]=freqz(b1,a1,N,'Whole');
H2.'
omega2.'
%%
```

The result:

Answer for problem (a):

$$\mathbf{a1} = [\,\mathbf{1} \quad -\mathbf{0.8}\,]$$

$$\mathbf{b1} = [\,\mathbf{2} \quad \mathbf{0} \quad -\mathbf{1}\,]$$

Answer for problem (b):

The output result:

```
  K>> H1.'

ans =

   5.0000 + 0.0000i   2.8200 - 1.3705i   1.8293 - 1.4634i   0.9258 - 0.9732i

  K>> omega1.'

ans =

         0    0.7854    1.5708    2.3562
```

Answer for problem (c):

The output result:

```
K>> H2.'

ans =

   5.0000 + 0.0000i   1.8293 - 1.4634i   0.5556 + 0.0000i   1.8293 + 1.4634i

K>> omega2.'

ans =

         0    1.5708    3.1416    4.7124
```

Summary and comments ：

I used the MATLAB to solve some problems such as creating and processing signals as well as plotting them.

I gained more impression of the concept of MATLAB and how it could be used in processing of the signals and systems. In this problem, I did some review on using the ‘a’ and ‘b’ to represent a system which is represented by the LCCDE. More than that, I also learned how to use the function ‘freqz’ to calculate the UIR of the system in the frequency domain. (When using the ‘freqz’ function, ‘N’ represents that the output frequency is on N’s equivalent division of π, when we add ‘whole’, ‘N’ represents that the output frequency is on N’s equivalent division of 2π)

Generally speaking, this problem let me understand the working principle of the ‘freqz’ function.

■ 3.4 Eigenfunctions of Discrete-Time LTI Systems

This exercise examines the eigenfunction property of discrete-time LTI systems. Complex exponentials are eigenfunctions of LTI systems, i.e., when the input sequence is a complex exponential, the output is the same complex exponential only scaled in amplitude by a complex constant. This constant can be computed from the impulse response $h[n]$. When the input to a discrete-time LTI system is $x[n] = z^n$, the output is $y[n] = H(z)z^n$, where

$$H(z) = \sum_{n=-\infty}^{\infty} h[n]z^{-n}.$$

Consider each of the following input signals:

$$\begin{aligned}
x_1[n] &= e^{j(\pi/4)n},\\
x_2[n] &= \sin(\pi n/8 + \pi/16),\\
x_3[n] &= (9/10)^n,\\
x_4[n] &= n + 1.
\end{aligned}$$

You will compute the outputs $y_1[n]$ through $y_4[n]$ that result when each of these signals is the input to the causal LTI system described by the linear, constant-coefficient difference equation

$$y[n] - 0.25y[n-1] = x[n] + 0.9x[n-1]. \tag{3.3}$$

Basic Problems

(a). Create a vector `n` containing the time indices for the interval $-20 \le n \le 100$ using the colon (`:`) operator. Using this vector, define `x1`, `x2`, `x3`, and `x4` to be vectors containing the values of the four signals $x_1[n]$ through $x_4[n]$ over the interval described by `n`. Produce a clearly labeled plot of each signal over this interval. Since the vector `x1` is complex, you will need to produce two separate plots for the real and imaginary parts. You can combine these in a single figure using either `subplot` or `hold`.

(b). The `filter` command described in Tutorial 2.2 computes the output of the causal, LTI system described by a difference equation for a given input sequence. Define `a` and `b` to specify the system shown in Eq. (3.3). Use these vectors and `filter` to compute the vectors `y1`, `y2`, `y3` and `y4`, containing the output of the system specified by Eq. (3.3) when the input is `x1` through `x4`, respectively. For each of the outputs you obtain, produce appropriately labeled plots of the portion of the outputs over the interval $0 \le n \le 100$. For `y1`, you will again need to plot the real and imaginary parts separately. Comparing your plots of inputs and outputs, indicate which of the input signals are eigenfunctions of this LTI system.

Note: In both this part and the next part we will ignore the output samples over the interval $-20 \le n \le -1$. These samples include transients due to the natural response of the system because MATLAB is unable to work with infinitely long input signals. MATLAB assumes the input and output were zero before the values given in `x`. The eigenfunction property of the system relates only to the steady-state solution. The signals and systems in this exercise have been chosen to insure the transients have died out completely within 20 samples, so by restricting the interval examined to be $0 \le n \le 100$, you are working with a portion of the output where the effect of the natural response is insignificant.

(c). For this part, you will verify which of the inputs were eigenfunctions and compute the corresponding eigenvalues for those eigenfunctions. If the vectors `x` and `y` describe the input and output sequences of a system, and the input sequence is an eigenfunction of the system, `y` should be equal to `x` scaled by a constant. This can be verified by computing `H=y./x`, which computes the ratio of the output to input sequence at each time index. If the resulting vector `H` is a constant, the input signal was an eigenfunction of the system. Compute `H1` through `H4` for each of the input/output signal pairs you obtained above, and produce appropriately labeled plots of `H` over the interval $0 \le n \le 100$ again. Again, `H1` will require separate plots for its real and imaginary parts. For the inputs which are eigenfunctions of the system, find the eigenvalue $H(z)$ from your plots or from the vector `H`.

The code:

```
%Name: Matlab: Signals and Systems Lab 5th
%Author: Changgang Zheng
%Student Number UESTC:2016200302027
%Student Number UoG:2289258z
%Institution: Glasgow College UESCT
%Question: Perform convolution. 3.4(a)(b)(c)

function  problem_2nd

    %% problem 3.4(a)
    n=[-20:100];
    x1=exp(1i*(pi/4).*n);
    x2=sin((pi/8).*n+pi/16);
    x3=(9/10).^n;
    x4=n+1;

    figure;                                                  % create a new
window for plotting
    subplot(4,1,1);
    stem(n,real(x1),'.');                                    % plot the
picture
    hold on;
    stem(n,imag(x1));                                        % plot the
picture
    title('The real part and imaginary part of x1[n]');      % name the
title of the figure as
    xlabel('n');                                             % name the
label of x-axis as 'n'
    ylabel('x1[n]');                                         % name the
label of y-axis as 'x1[n]'
    legend('Real part of x1[n]','imaginary part of x1[n]'); % set the label
for each line 'Real part of x1[n]','imaginary part of x1[n]'
    grid on;


    subplot(4,1,2);                                          % set the place
for pictures in a picture, which is plotted in a window
    stem(n,x2,'.');                                          % plot the
picture
    title('The graph of x2[n]');                             % name the
title of the figure as 'The graph of x2[n]'
    xlabel('n');                                             % name the
label of x-axis as 'n'
    ylabel('x2[n]');                                         % name the
label of y-axis as 'x2[n]'
    grid on;

    subplot(4,1,3);                                          % set the place
for pictures in a picture, which is plotted in a window
    stem(n,x3,'.');                                          % plot the
picture
    title('The graph of x3[n]');                             % name the
title of the figure as
    xlabel('n');                                             % name the
label of x-axis as 'n'
    ylabel('x3[n]');                                         % name the
label of y-axis as 'x3[n]'
```

```
    grid on;

    subplot(4,1,4);                                         % set the place
for pictures in a picture, which is plotted in a window
    stem(n,x4,'.');                                         % plot the
picture
    title('The graph of x4[n]');                            % name the
title of the figure as
    xlabel('n');                                            % name the
label of x-axis as 'n'
    ylabel('x4[n]');                                        % name the
label of y-axis as 'x4[n]'
    grid on;


    %%

    %% problem 3.4(b)
    a=[1 -0.25];
    b=[1 0.9];
    y1=filter(b,a,x1);
    y2=filter(b,a,x2);
    y3=filter(b,a,x3);
    y4=filter(b,a,x4);

    figure;                                                 % create a new
window for plotting
    subplot(4,1,1);

    stem([-20:-1],real(y1(1:20)),'.','g');                  % plot the
picture
    hold on;
    stem([0:100],real(y1(21:121)),'.');                     % plot the
picture
    hold on;
    stem([-20:-1],imag(y1(1:20)),'.','m');                  % plot the
picture
    hold on;
    stem([0:100],imag(y1(21:121)),'.');                     % plot the
picture
    title('the imaginary part and the real part of the response of y1[n]');
    xlabel('n');                                            % name the
label of x-axis as 'n'
    ylabel('y1[n](imaginary&real)');                        % name the
label of y-axis as 'y1[n](imaginary) and y1[n](real)'
    legend('real part of y1[n]','real part of y1[n]','imaginary part of
y1[n]','imaginary part of y1[n]')
    grid on;

                                                            % create a new
window for plotting
    subplot(4,1,2);                                         % set the place
for pictures in a picture, which is plotted in a window
    stem([-20:-1],y2(1:20),'.');                            % plot the
picture
    hold on;
    stem([0:100],y2(21:121),'.');                           % plot the
picture
    title('y2[n]----the response of x2[n]');                % name the
title of the figure as 'y2[n]----the response of x2[n]'
```

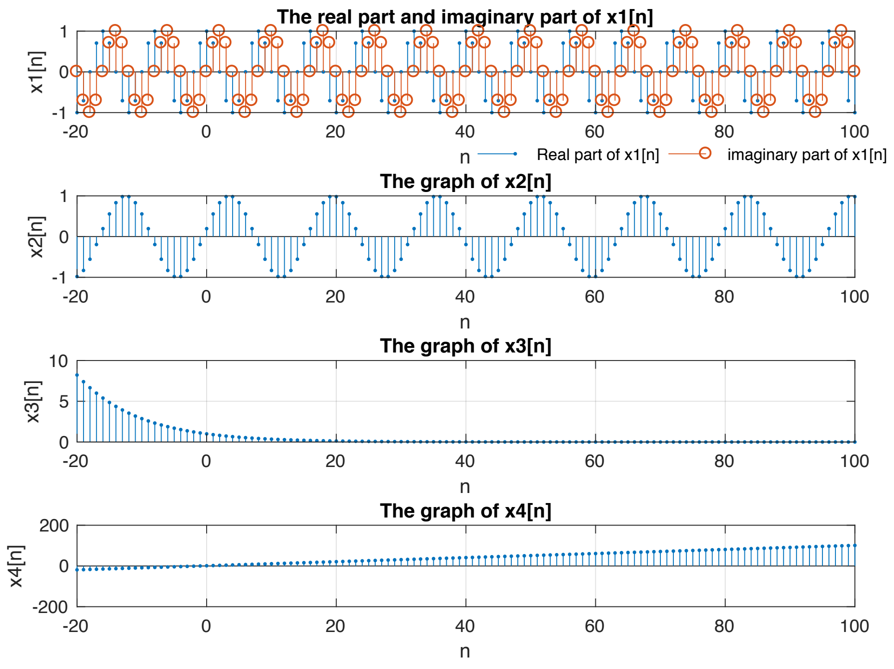
```
    xlabel('n');                                            % name the
label of x-axis as 'n'
    ylabel('y2[n]');                                        % name the
label of x-axis as 'y2[n]'
    grid on;

    subplot(4,1,3);                                         % set the place
for pictures in a picture, which is plotted in a window
     stem([-20:-1],y3(1:20),'.');                           % plot the
picture
    hold on;
    stem([0:100],y3(21:121),'.');                           % plot the
picture                                             % plot the picture
    title('y3[n]----the response of x3[n]');                % name the
title of the figure as
    xlabel('n');                                            % name the
label of x-axis as 'n'
    ylabel('y3[n]');                                        % name the
label of x-axis as 'y3[n]'
    grid on;

    subplot(4,1,4);                                         % set the place
for pictures in a picture, which is plotted in a window
     stem([-20:-1],y4(1:20),'.');                           % plot the
picture
    hold on;
    stem([0:100],y4(21:121),'.');                           % plot the
picture
    title('y4[n]----the response of x4[n]');                % name the
title of the figure as
    xlabel('n');                                            % name the
label of x-axis as 'n'
    ylabel('y4[n]');                                        % name the
label of x-axis as 'y4[n]'
    grid on;

    %%

    %% problem 3.4(c)
    H1=y1./x1;
    H2=y2./x2;
    H3=y3./x3;
    H4=y4./x4;
    % H1_real=real(y1)./real(x1); % wrong
    % H1_imag=imag(y1)./imag(x1); % wrong
    H1_real=real(y1./x1);
    H1_imag=imag(y1./x1);


    figure;                                                 % create a new
window for plotting
    subplot(4,1,1);                                         % set the place
for pictures in a picture, which is plotted in a window
    stem([-20:-1],H1_real(1:20),'.','g');                   % plot the
picture
    hold on;
    stem([0:100],H1_real(21:121),'.');                      % plot the
picture
    hold on;
    stem([-20:-1],H1_imag(1:20),'.','m');                   % plot the
```

```
picture
    hold on;
    stem([0:100],H1_imag(21:121),'.');                     % plot the
picture
    title('real & imaginary part of H1');                  % name the
title of the figure as
    xlabel('n');                                           % name the
label of x-axis as 'n'
    ylabel('H1-real & H1-imag[n]');                        % name the
label of x-axis as 'H1_imag[n]'
    legend('real part of H1[n]','real part of H1[n]','imaginary part of
H1[n]','imaginary part of H1[n]')
    grid on;

    subplot(4,1,2);                                        % set the place
for pictures in a picture, which is plotted in a window
    stem([-20:-1],H2(1:20),'.');                           % plot the
picture
    hold on;
    stem([0:100],H2(21:121),'.');                          % plot the
picture
    title('H2[n]');                                        % set the title
    xlabel('n');                                           % name the
label of x-axis as 'n'
    ylabel('H2[n]');                                       % name the
label of y-axis as H2[n]
    grid on;

    subplot(4,1,3);                                        % set the place
for pictures in a picture, which is plotted in a window
    stem([-20:-1],H3(1:20),'.');                           % plot the
picture
    hold on;
    stem([0:100],H3(21:121),'.');                          % plot the
picture
    title('H3[n]');                                        % set the title
    xlabel('n');                                           % name the
label of x-axis as 'n'
    ylabel('H3[n]');                                       % name the
label of y-axis as H3[n]
    grid on;

    subplot(4,1,4);                                        % set the place
for pictures in a picture, which is plotted in a window
    stem([-20:-1],H4(1:20),'.');                           % plot the
picture
    hold on;
    stem([0:100],H4(21:121),'.');                          % plot the
picture
    title('H4[n]');                                        % set the title
    xlabel('n');                                           % name the
label of x-axis as 'n'
    ylabel('H4[n]');                                       % name the
label of y-axis as H4[n]
    grid on;

    %%
```

The result:

Answer for problem (a):

The result of the plot:

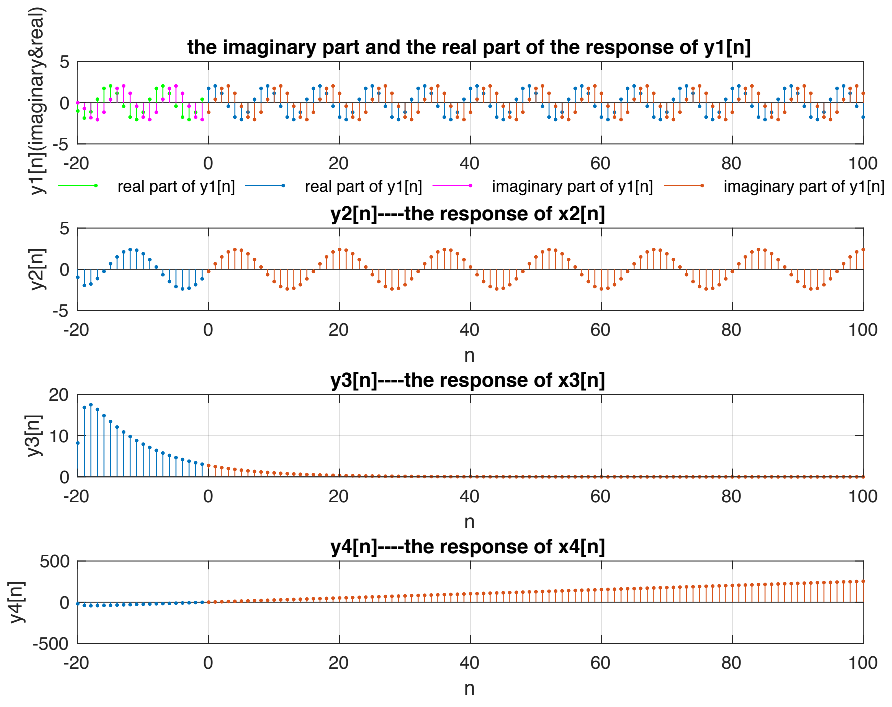


The input signal of the system, which is represented by the LCCDE.

Answer for problem (b):

I also plot the result from -20 to -1, which is not necessary from the question. However, from this, we can see what happens in these area.

The result of the plot:



The output signal of the system, which is represented by the LCCDE.

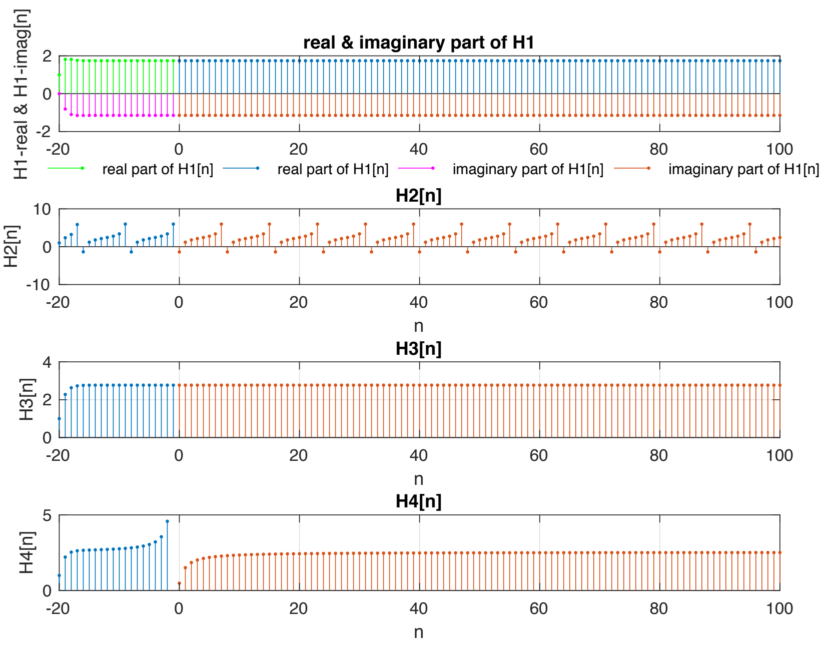
$$Output = EigenValue \times EigenFunction$$

If compare the input signal and the output signal directly, I suppose that the signal 3, which can also be represented by the x3[n] could be the eigenfunction of the system, as all the output is the multiplication of the input.

Answer for problem (c):

I also plot the result from -20 to -1, which is not necessary from the question. Because from this, we can see what happens in these area clearly.

The result of the plot:



The output signal of the system divides by the input signal, which is used to check which input is the eigenfunction.

The signal H[n] is get from y[n]/x[n], we also know that the eigenvalue is only an integer, so when the input is the eigenfunction, the signal H should be the eigenvalue, which means that all the number should be the same in the interval $0 \leq n \leq 100$. Only the H3[n] meets this need, so the x3[n] is the eigenfunction of this system and that integer (2.7692) is the eigenfunction.

Summary and comments：

I used the MATLAB to solve some problems such as creating and processing signals as well as plotting them.

I gained more impression of the concept of MATLAB and how it could be used in processing of the signals and systems. I earned a deep impression on what is the eigenfunction and what is the eigenvalue.

$$Output = EigenValue \times EigenFunction$$

If all the output of the system divides the input equals to a same integer, the input is the eigenfunction and the integer is the eigenvalue. Also, it is interesting to know that when we use the filter, some integers at the beginning of the output should be ignored as the value of them remain oscillation (which is different from the real condition) (caused by the flaw of Matlab).

■ 3.5 Synthesizing Signals with the Discrete-Time Fourier Series

The discrete-time Fourier series (DTFS) is a frequency-domain representation for periodic discrete-time sequences. The synthesis and analysis equations for the DTFS are given by Eqs. (3.1) and (3.2). This exercise contains three sets of problems to give you practice working with these equations. The Basic Problems allow you to synthesize a very simple periodic discrete-time signal from its DTFS coefficients. In the Intermediate Problems, you will both analyze a set of periodic discrete-time signals to obtain their DTFS coefficients, and construct one of these signals by adding in a few coefficients at a time. For the Advanced Problem, you will write a function to find the DTFS coefficients of an arbitrary periodic discrete-time signal from one period of samples.

Intermediate Problems

For these problems, you will examine the DTFS representation of several different square waves. Specifically, you will look at signals

$$x_1[n] = \begin{cases} 1, & 0 \le n \le 7, \end{cases} \tag{3.4}$$

$$x_2[n] = \begin{cases} 1, & 0 \le n \le 7, \\ 0, & 8 \le n \le 15, \end{cases} \tag{3.5}$$

$$x_3[n] = \begin{cases} 1, & 0 \le n \le 7, \\ 0, & 8 \le n \le 31, \end{cases} \tag{3.6}$$

where $x_1[n]$, $x_2[n]$ and $x_3[n]$ have periods of $N_1 = 8$, $N_2 = 16$ and $N_3 = 32$, respectively.

(d). Define three vectors `x1`, `x2`, and `x3` representing one period of each of the signals $x_1[n]$, $x_2[n]$ and $x_3[n]$. Using these vectors, make appropriately labeled plots of each of the signals over the range $0 \le n \le 63$. Note: You will have to repeat the vectors you have defined to cover this range of samples.

(e). Exercise 3.1 explains how to use `fft` to compute the DTFS of a periodic discrete-time signal from one period of the signal. Using the `fft` function, define the vectors `a1`, `a2`, and `a3` to be the DTFS coefficients of $x_1[n]$ through $x_3[n]$, respectively. Generate appropriately labeled plots of the magnitude of each of the DTFS coefficient sequences using `abs` and `stem`. From your time domain plots of Part (d) and Eq. (3.2), you should be able to predict the values of `a1(1)`, `a2(1)` and `a3(1)` — the DC components of the signals. Do your predicted values match those obtained with MATLAB?

(f). In this part, you will examine how $x_3[n]$ appears when it is synthesized a few coefficients at a time. Using the vector `a3` you found in the previous part, define vectors `x3_2`, `x3_8`, `x3_12` and `x3_all` corresponding to the following four signals

$$x_{3\_2}[n] = \sum_{k=-2}^{2} a_k e^{jk(2\pi/32)n},$$

$$x_{3\_8}[n] = \sum_{k=-8}^{8} a_k e^{jk(2\pi/32)n},$$

$$x_{3\_12}[n] = \sum_{k=-12}^{12} a_k e^{jk(2\pi/32)n},$$

$$x_{3\_\mathrm{all}}[n] = \sum_{k=-15}^{16} a_k e^{jk(2\pi/32)n},$$

on the interval $0 \le n \le 31$. Note that since $x_3[n]$ is real, the DTFS coefficients for this signal will be conjugate symmetric, i.e., $a_k = a^*_{-k}$. Because a_k is conjugate symmetric and all of the sums except $x_{3\_\mathrm{all}}[n]$ are symmetric about $k = 0$, the resulting time signals should be purely real. If you are unclear about why this is true, you may want to review the symmetry properties of the DTFS. Due to roundoff error in MATLAB, you may need to discard some very small imaginary parts of the signals you synthesize using `real`. The sums specified above are symmetric about $k = 0$ but the vector `a3` represents a_k for $k = 0, \ldots, 31$ as [`a3(1)`, ... , `a3(32)`], so you will need to determine which elements of `a` correspond to the negative values of k when you implement the sums.

(h). Generate a sequence of appropriately labeled plots using `stem` showing how the signals you created converge to $x_3[n]$ as more of the DTFS coefficients are included in the sum. Specifically, `x3_all` should be equal to the original vector `x3` within the bounds of MATLAB's roundoff error. Does the synthesis of this discrete-time square wave display the Gibb's phenomenon?

The code:

```
%Name: Matlab: Signals and Systems Lab 5th
%Author: Changgang Zheng
%Student Number UESTC:2016200302027
%Student Number UoG:2289258z
%Institution: Glasgow College UESCT
%Question: Perform convolution. 3.5(d)(e)(f)(h)

function  problem_3rd

    %% problem 3.5(d)
    N=64;
    n=[0:63];
    X1=[ones(1,8)];
    X2=[ones(1,8) zeros(1,8)];
    X3=[ones(1,8) zeros(1,24)];
    x1=zeros(1,64);
    x2=zeros(1,64);
    x3=zeros(1,64);
    N1=8;
    N2=16;
    N3=32;

    for i=1:N
        n1=mod(i-1,N1)+1;
        x1(i)=X1(n1);
        n2=mod(i-1,N2)+1;
        x2(i)=X2(n2);
        n3=mod(i-1,N3)+1;
        x3(i)=X3(n3);
    end

    figure;                                                    % create a new
window for plotting
    subplot(3,1,1);                                            % set the place
for pictures in a picture, which is plotted in a window
    stem(n,x1,'.');                                            % plot the
picture
    title('The graph of x1[n]');                               % name the
title of the figure as'The graph of x1[n]'
    xlabel('n');                                               % name the
label of x-axis as 'n'
    ylabel('x1[n]');                                           % name the
label of y-axis as 'x1[n]'
    grid on;

    subplot(3,1,2);                                            % set the place
for pictures in a picture, which is plotted in a window
    stem(n,x2,'.');                                            % plot the
```

```
picture
    title('The graph of x2[n]');                            % name the
title of the figure as 'The graph of x2[n]'
    xlabel('n');                                            % name the
label of x-axis as 'n'
    ylabel('x2[n]');                                        % name the
label of y-axis as 'x2[n]'
    grid on;

    subplot(3,1,3);                                         % set the place
for pictures in a picture, which is plotted in a window
    stem(n,x3,'.');                                         % plot the
picture
    title('The graph of x3[n]');                            % name the
title of the figure as 'The graph of x3[n]'
    xlabel('n');                                            % name the
label of x-axis as 'n'
    ylabel('x3[n]');                                        % name the
label of y-axis as 'x3[n]'
    grid on;
    %%
    %% problem 3.5(e)

    a1=(1/N1)*fft(x1(1:8));
    a2=(1/N2)*fft(x2(1:16));
    a3=(1/N3)*fft(x3(1:32));

    figure;                                                 % create a new
window for plotting
    subplot(3,1,1);                                         % set the place
for pictures in a picture, which is plotted in a window
    stem([0:7],abs(a1(1:8)),'.');                           % plot the
picture
    title('a1[k]');
    xlabel('k');                                            % name the
label x
    ylabel('a1[k]');                                        % name the
label y
    grid on;                                                % generate the
grid

    subplot(3,1,2);                                         % set the place
for pictures in a picture, which is plotted in a window
    stem([0:15],abs(a2(1:16)),'.');                         % plot the
picture
    title('x2[k]');
    xlabel('k');                                            % name the
label x
    ylabel('a2[k]');                                        % name the
label y
    grid on;                                                % generate the
grid

    subplot(3,1,3);                                         % set the place
for pictures in a picture, which is plotted in a window
    stem([0:31],abs(a3(1:32)),'.');                         % plot the
picture
    title('a3[k]');
    xlabel('k');                                            % name the
label x
    ylabel('a3[k]');                                        % name the
```

```
label y
    grid on;

    a1=shift_fft(a1);
    a2=shift_fft(a2);
    a3=shift_fft(a3);

    %{
    A1=a1;
    A2=a2;
    A3=a3;

    for i=1:length(a1)/2-1
        a1(length(a1)/2)=A1(1);
        a1(length(a1)/2+i)=A1(1+i);
        a1(length(a1)/2-i)=A1(length(a1)-i+1);
        a2(length(a2)/2)=A2(1);
        a2(length(a2)/2+i)=A2(1+i);
        a2(length(a2)/2-i)=A2(length(a2)-i+1);
        a3(length(a3)/2)=A3(1);
        a3(length(a3)/2+i)=A3(1+i);
        a3(length(a3)/2-i)=A3(length(a3)-i+1);
    end
    %}

    figure;                                                % create a
new window for plotting
    subplot(4,1,1);                                        % set the
place for pictures in a picture, which is plotted in a window
    stem([0:7],x1(1:8),'.');                               % plot the
picture
    title('x1[n]');
    xlabel('n');                                           % name the
label x
    ylabel('x1');                                          % name the
label y
    grid on;                                               % generate
the grid

    subplot(4,1,2);                                        % set the
place for pictures in a picture, which is plotted in a window
    stem([-floor(length(a1)/2)+1:length(a1)/2],a1,'.');    % plot the
picture
    title('a[k] of x1');
    xlabel('k');                                           % name the
label x
    ylabel('a[k]');                                        % name the
label y
    grid on;                                               % generate
the grid

    subplot(4,1,3);                                        % set the
place for pictures in a picture, which is plotted in a window
    stem([-floor(length(a1)/2)+1:length(a1)/2],abs(a1),'.');    % plot the
picture
    title('Amplitude of a[k]');
    xlabel('k');                                           % name the
label x
    ylabel('|a[k]|');                                      % name the
label y
```

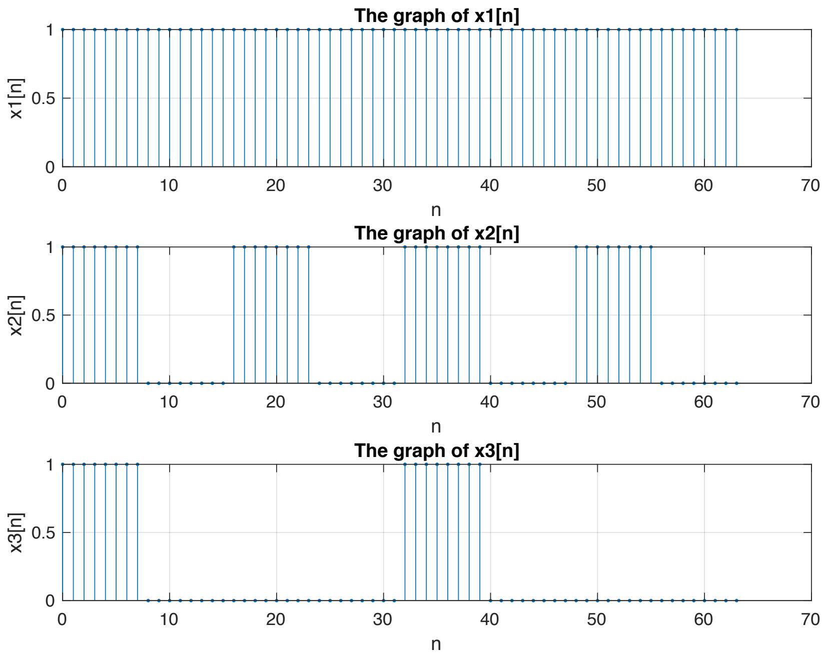
```
    grid on;                                                 % generate 
the grid

    subplot(4,1,4);                                          % set the 
place for pictures in a picture, which is plotted in a window
    stem([-floor(length(a1)/2)+1:length(a1)/2],phase(a1),'.');  % plot the 
picture
    title('Phase of a[k]');                                  % set the 
title
    xlabel('k');                                             % name the 
label x
    ylabel('Phase of a[k]');                                 % name the 
label y
    grid on;                                                 % generate 
the grid

    figure;                                                  % create a 
new window for plotting
    subplot(4,1,1);                                          % set the 
place for pictures in a picture, which is plotted in a window
    stem([0:15],x2(1:16),'.');                               % plot the 
picture
    title('x2[n]');
    xlabel('n');                                             % name the 
label x
    ylabel('x2[n]');                                         % name the 
label y
    grid on;                                                 % generate 
the grid

    subplot(4,1,2);                                          % set the 
place for pictures in a picture, which is plotted in a window
    stem([-floor(length(a2)/2)+1:length(a2)/2],a2,'.');      % plot the 
picture
    title('a[k] of x2');                                     % set the 
title
    xlabel('k');                                             % name the 
label x
    ylabel('a[k]');                                          % name the 
label y
    grid on;                                                 % generate 
the grid

    subplot(4,1,3);                                          % set the 
place for pictures in a picture, which is plotted in a window
    stem([-floor(length(a2)/2)+1:length(a2)/2],abs(a2),'.');  % plot the 
picture
    title('Amplitude of a[k]');
    xlabel('k');                                             % name the 
label x
    ylabel('|a[k]|');                                        % name the 
label y
    grid on;                                                 % generate 
the grid

    subplot(4,1,4);                                          % set the 
place for pictures in a picture, which is plotted in a window
    stem([-floor(length(a2)/2)+1:length(a2)/2],phase(a2),'.');  % plot the 
picture
    title('Phase of a[k]');
    xlabel('k');                                              % name the
label x
    ylabel('Phase of a[k]');                                  % name the
label y
    grid on;                                                  % generate
the grid

    figure;                                                   % create a
new window for plotting
    subplot(4,1,1);                                           % set the
place for pictures in a picture, which is plotted in a window
    stem([0:31],x3(1:32),'.');                                % plot the
picture
    title('x3[n]');                                           % set the
title
    xlabel('k');                                              % name the
label x
    ylabel('x3[n]');                                          % name the
label y
    grid on;                                                  % generate
the grid

    subplot(4,1,2);                                           % set the
place for pictures in a picture, which is plotted in a window
    stem([-floor(length(a3)/2)+1:length(a3)/2],a3,'.');       % plot the
picture
    title('a[k] of x3');                                      % set the
title
    xlabel('k');                                              % name the
label x
    ylabel('a[k]');                                           % name the
label y
    grid on;                                                  % generate
the grid

    subplot(4,1,3);                                           % set the
place for pictures in a picture, which is plotted in a window
    stem([-floor(length(a3)/2)+1:length(a3)/2],abs(a3),'.');  % plot the
picture
    title('Amplitude of a[k]');                               % set the
title
    xlabel('k');                                              % name the
label x
    ylabel('|a[k]|');                                         % name the
label y
    grid on;                                                  % generate
the grid

    subplot(4,1,4);                                           % set the
place for pictures in a picture, which is plotted in a window
    stem([-floor(length(a3)/2)+1:length(a3)/2],phase(a3),'.'); % plot the
picture
    title('Phase of a[k]');                                   % set the
title
    xlabel('k');                                              % name the
label x
    ylabel('Phase of a[k]');                                  % name the
label y
    grid on;                                                  % generate
the grid
```

```
    %%
    %% problem 3.5(f)
    syms k n;
    E=@(k,n)(exp(k*1i*(2*pi/32)*n));

    x3_2=zeros(1,32);
    x3_8=zeros(1,32);
    x3_12=zeros(1,32);
    x3_all=zeros(1,32);

    for n=0:31
        for k=-2:2
            x3_2(n+1)=real(x3_2(n+1)+E(k,n)*a3(16+k));
        end
        for k=-8:8
            x3_8(n+1)=real(x3_8(n+1)+E(k,n)*a3(16+k));
        end
        for k=-12:12
            x3_12(n+1)=real(x3_12(n+1)+E(k,n)*a3(16+k));
        end
        for k=-15:16
            x3_all(n+1)=real(x3_all(n+1)+E(k,n)*a3(16+k));
        end
    end

    %%
    %% problem 3.5(h)

    figure;                                                 % create a new
window for plotting
    subplot(4,1,1);                                         % set the place
for pictures in a picture, which is plotted in a window
    stem([0:length(x3_2)-1],x3_2,'.');                      % plot the
picture
    title('x3-2[n]');                                       % set the title
    xlabel('n');                                            % name the
label x
    ylabel('x3-2[n]');                                      % name the
label y
    grid on;                                                % generate the
grid

    subplot(4,1,2);                                         % set the place
for pictures in a picture, which is plotted in a window
    stem([0:length(x3_8)-1],x3_8,'.');                      % plot the
picture
    title('x3-8[n]');                                       % set the title
    xlabel('n');                                            % name the
label x
    ylabel('x3-8[n]');                                      % name the
label y
    grid on;                                                % generate the
grid

    subplot(4,1,3);                                         % set the place
for pictures in a picture, which is plotted in a window
    stem([0:length(x3_12)-1],x3_12,'.');                    % plot the
picture
    title('x3-12[n]');                                      % set the title
    xlabel('n');                                          % name the
label x
    ylabel('x3-12[n]');                                   % name the
label y
    grid on;                                              % generate the
grid

    subplot(4,1,4);                                       % set the place
for pictures in a picture, which is plotted in a window
    stem([0:length(x3_all)-1],x3_all,'.');                % plot the
picture
    title('x3-all[n]');                                   % set the title
    xlabel('n');                                          % name the
label x
    ylabel('x3-all[n]');                                  % name the
label y
    grid on;                                              % generate the
grid

    %%
```

The result:

Answer for problem (d):

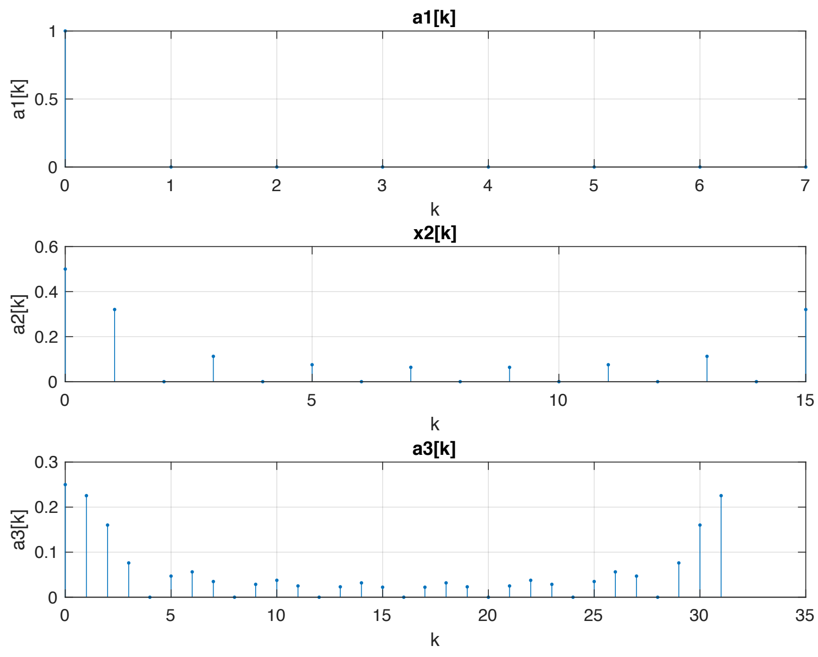
The result of the plot



The output of the input signals

Answer for problem (e):

We just select the input signal in an interval and use the function 'fft' to get the output, which could be plot as follows. But these graphs are different from the real condition.
The result of the plot:



The output of the a1[k] to a3[k].

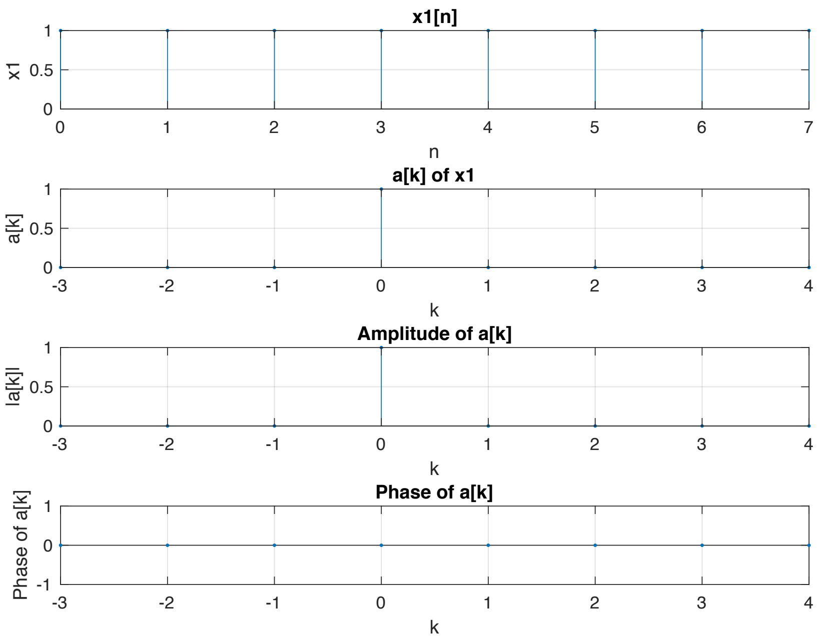
The reason why these graphs are different from the real condition is because of the function 'fft'. The output is out of order, that the first output is the DC component, which is followed by the a_k with k from k=1 to k=N and with k from k=-N to k=-1 (Totally 2N+1 points, which is the same as input point numbers). When the graphs are plotted, the fist output is plotted at the k=0 point, which is also the DC component. It should be equal to the real estimate.

As the output from function 'fft' is in a wrong order, I defined a function 'shift_fft' (thus we can solve the problem easier, as we don't need to calculate the conjugation any more by using it) which is similar as the 'fftshift'. The different is that if the input is the even number, the rest point is added to the right side ('fft' is to the left).

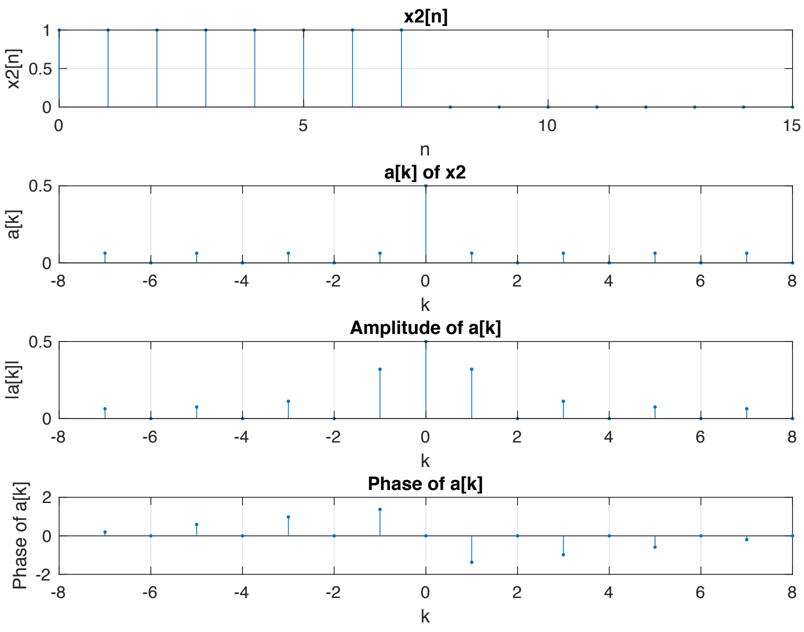
Code:

```
%Name: Matlab/CUDA: Signals and Systems Lab 5th
%Auther: Changgang Zheng
%Student Number UESTC:2016200302027
%Student Number UoG:2289258z
%Institution: Glasgow College UESCT

function  a=shift_fft(a)
    a=[a(floor((length(a)/2))+2:length(a)) a(1:floor(length(a)/2)+1)];
```

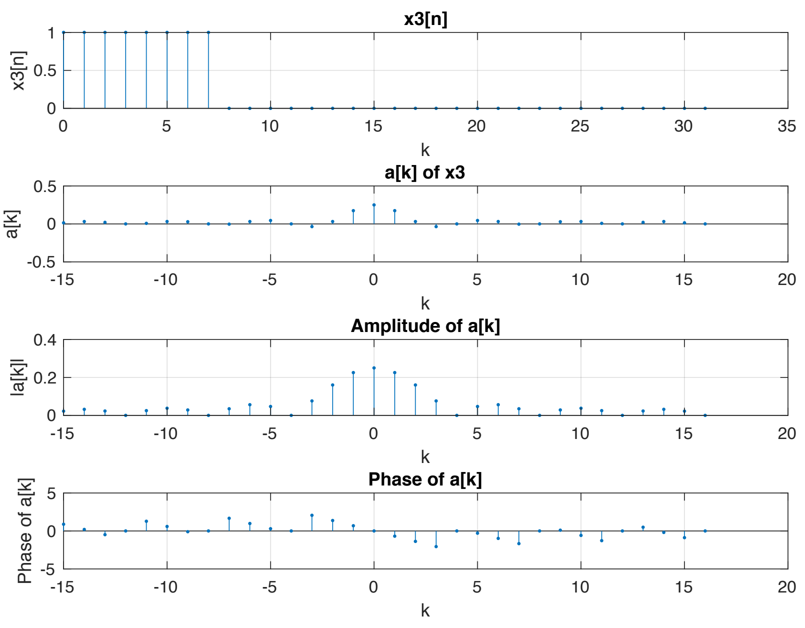
The following three series of graphs are about the property of the a[k]. In each series, the first one is the input signal in the time domain, the second is the a[k] getting from the function 'fft' and 'shift_fft', the third one is the amplitude of the a[k] (which is the same as |a[k]|), and the last picture in each series is about the phase of the each a[k].



The x1[n], the a1[k], |a1[k]| and phase a1[k].



The x2[n], the a2[k], |a2[k]| and phase a2[k].

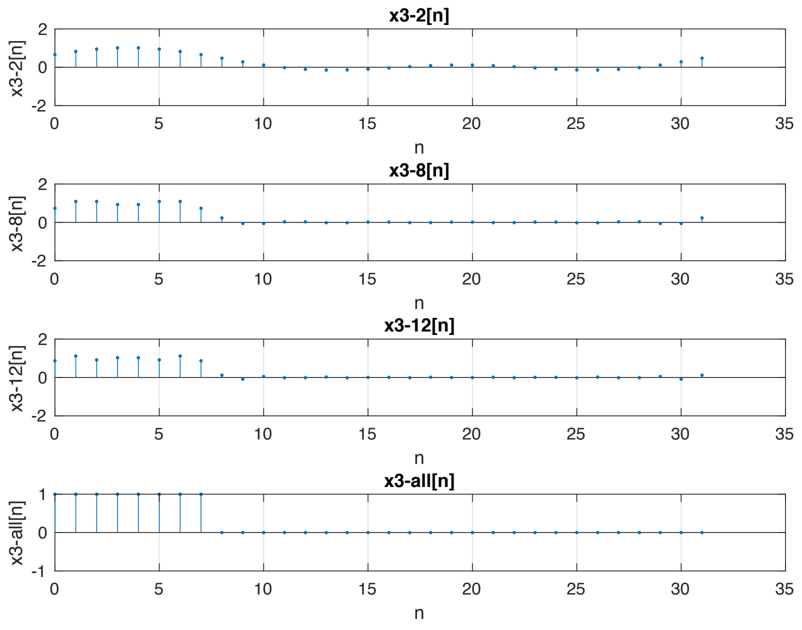


The x3[n], the a3[k], |a3[k]| and phase a3[k].

Answer for problem (f):

Because I have already used the function 'shift_fft' (defined by myself) to reset the order of a[k], so I did not need to use the property of the Fourier series like $a_k = a^*_{-k}$ as I can find all of the necessary a_k directly. (detail can be seen in the above code)

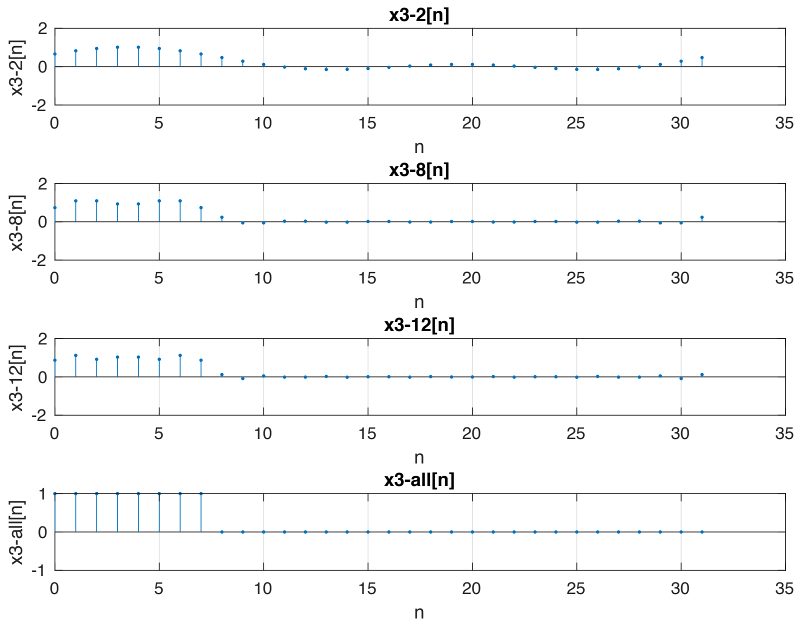
The result of the plot:



Answer for problem (h):

When we do the inverse, with the increase of the numbers of the range of the k, the fitting result is getting better and better.

The result of the plot:



According to this x3-all[n], this discrete time square wave did not display the Gibb's phenomena, as the peak of the edge is the same as the real condition (input x3[n]).

Summary and comments：

I used the MATLAB to solve some problems such as creating and processing signals as well as plotting them.

I gained more impression of the concept of MATLAB and how it could be used in processing of the signals and systems, especially on the part of the Fourier series. I understood how to use the Matlab to calculate the Fourier series by using the 'fft' function and understood that the output of this function is not in the right order. More than that, we can use the function 'fftshift' or the function 'shift_fft' (defined by me) to shift the result and make it in the right order.

I also understood the nature of the a[k] which can be represented in the polar form and includes two essential parts: |a[k]| and phase of the a[k], which can get from function ‘abs’ and ‘phase’ respectively. Amplitude |a[k]| is the gain and the phase is the phase shift when we use the H[jw] to represent the LTI system.

Moreover, when we inverse the signal from the Fourier series to the time domain, with the increase of the numbers of the range of the k, the output fitting result would become better and better.

Suggestion for this lab：

It would be better if we can get the answers of the lab to check if our work is staying on track. Thanks!

Score：

Instructor：