UNIVERSITY OF LIVERPOOL

Experiment 3 - Fourier Synthesis of Periodic Waveforms

Department of Electrical Engineering & Electronics

November 26, 2021

Abstract

The methods of synthesising Fourier series are applied practically through Matlab. And the experiment verifies that the periodic waveforms can be synthesised by the sum of sinusoids which have frequencies harmonically. Moreover, the effects of Low-pass and Band-pass filter with different phase shift are both discussed and summarised. Although different filters are applied with different phase shift, the fundamental frequency will not be changed. Additionally, the percentage overshoot is calculated in different harmonic value, which is also compared with the expected limit 17.9%. Meanwhile, the effects of phase shift in one sound and the whole chord are both analysed with providing the specific reasons that why humans are hard to distinguish the differences between two sounds with phase shift. In different question parts, the same part of the code (initialization part, plotting waveform part) may not be repeatedly displayed, only the important algorithm part of the code will be displayed. The whole codes of each part can be viewed in the appendix.

Declaration

I confirm that I have read and understood the University's definitions of plagiarism and collusion from the Code of Practice on Assessment. I confirm that I have neither committed plagiarism in the completion of this work nor have I colluded with any other party in the preparation and production of this work. The work presented here is my own and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web). I understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment.

Contents

1 Introduction

In mathematics, Fourier analysis is the study of how general functions can be expressed or approximated by the sum of simpler trigonometric functions [1]. These trigonometric function are mostly sines and cosines(possibly plus a constant). If periodic signal f(t) is expressed as the sum of sine and cosine functions with period T, then it can be called a Fourier series with some constants known as Fourier coefficients [2].

1.1 Theoretical Background

1.1.1 Waveform background

A signal with time-varying or spatial-varying will generally be used in the process of signal processing. Especially, the most familiar signal used is the sine wave shown in the following figure [2].

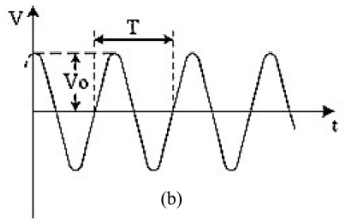


Figure 1: Sinusoid waveform (taken from [2])

If voltage (V) and time (t) are provided, then the equation can be expressed by the following formula:

$$V(t) = V_0 \sin(2 * \pi * f * t + \phi) \tag{1}$$

Where V_0 is the amplitude, f is the frequency and ϕ is the phase angle. In addition, if the equation is a period function , then (for all integer t):

$$V(t + n * T) = V(t) \tag{2}$$

The unit for frequency is KHz or Hz and the period T is equal to 1/f. The phase angle will determine the starting point when t=0 [2].

1.1.2 Fourier analysis background

Many signals in real life are periodic but non-sinusoidal. Therefore, Fourier analysis can be used to simulate these irregular signals. The Fourier analysis depends on the theorem that a periodic waveform can be expressed as the sum of infinite sine waves or cosine waves adding a constant possibly. The following is the common formula for Fourier analysis:

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n * cos(2 * \pi * n * f_0 * t) + \sum_{n=1}^{\infty} b_n * sin(2 * \pi * n * f_0 * t) \tag{3}$$

Where f_0 is the fundamental frequency and a_0,a_n,b_n are the calculated Fourier coefficients. The specific calculation can be seen in the following equations:

$$a_0 = \frac{1}{T}\int_0^T f(t)dt \tag{4}$$

$$a_n = \frac{2}{T}\int_0^T f(t) * cos(2 * \pi * n * f_0 * t)dt \tag{5}$$

$$b_n = \frac{2}{T}\int_0^T f(t) * sin(2 * \pi * n * f_0 * t)dt \tag{6}$$

Additionally, the alternative form for equation 3 is:

$$f(t) = c_0 + \sum_{n=1}^{\infty} c_n * cos(2 * \pi * n * f_0 * t + \phi_n) \tag{7}$$

Where $c_n = \sqrt{(a_n)^2 + (b_n)^2}$ and $tan(\phi_n) = -b_n/a_n$

Because f(t) is expressed by the sum of DC voltage and sinusoids of different frequencies, the first frequency is called the fundamental frequency and other sinusoids are called harmonic components.

1.1.3 Synthesis background

To synthesize a periodic wave with a good approximation, it is necessary to generate sine and cosine waves with appropriate amplitudes (defined by coefficients) and combine them to the highest possible value of n [2]. The larger the value of n for generating sine and cosine wave signals, the closer the synthesized waveform is to the original waveform. More specific examples can be viewed in the lab script, the following is an example of sawtooth signal.

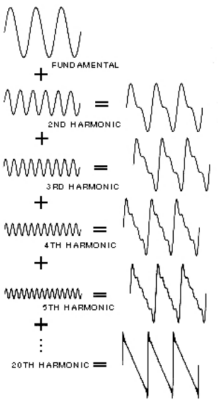


Figure 2: Approximation of sawtooth signal (taken from [2])

From the above example, it can be seen clearly that as the value of n increases, the **rippled** waveform gradually becomes a **triangular** wave. This also verifies the above conclusion.

1.2 Objectives

The main purpose of this experimental is to familiarize with the Fourier analysis. The specific objectives can be divided into the following aspects:

1. Quantitatively study the Fourier components of some basic periodic waveforms.
2. Verify whether it is possible to synthesize periodic waveforms by the sum of sinusoids
3. Investigate the effect of the filter on the waveform including the low-pass and band-pass filters with phase response.
4. Study the effect of phase on sound.

1.3 Apparatus

In this experiment, Matlab will be used to synthesis arbitrary waveforms using harmonics. MATLAB is a business mathematics software produced by MathWorks. It is used in data analysis, wireless communication, deep learning, image processing and computer vision, **signal processing**, quantitative finance and risk management, robotics, control systems and other fields [3]. For each different task, the corresponding code will be provided in the appendix.

2 Results and discussion

In this part, the specific results for each part will be discussed including codes and resulting waveforms. Additionally, all questions will be answered and the explanations will also be offered.

2.1 Part A (30 marks)

2.1.1 Part A requirements

In part A, the following function is required to synthesise the waveform in different harmonics. In addition, the spectrum (frequency domain view) of f(t) is also required using Matlab code.

$$f(t) = V_0(sin(\omega * t) + \frac{1}{3} * sin(3 * \omega * t) + \frac{1}{5} * sin(5 * \omega * t) + \frac{1}{7} * sin(7 * \omega * t) + \frac{1}{9} * sin(9 * \omega * t)) \quad (8)$$

Where $V_0 = 1V$ and $w = 2 * \pi * f_0$

2.1.2 Answer the questions

In the next section, each question will be answered in the order of the questions provided in the template. The whole codes will be provided in the appendix according to different parts.

- **Question A.1**

  Provide the Matlab code required to synthesise the waveform in equation 12 and the resulting waveform. [5 marks]

In this question, the first 10 harmonics of this waveform will be synthesized. Since the codes are divided into three parts, the specific code for each part is as follows:

1. **Define the variables and initialize the function**

```
%Assume f0 = 1000Hz and V0=1V
f0 = 1000;
V0=1;

t = 0:0.00001:0.001; %Define the time variable
harmonics =10; %In part A, the harmonic will change
ft_final = 0; %Initialize the function
```

Figure 3: Define the variables and initialize the function

It can be seen from the above variables: fundamental frequency and V_0 are assumed as 1000Hz and 1V. And the time variable and harmonic value are also set in this part. Finally, the value of the function will be initialized to 0. In the part A, the value of harmonic will be changed for observing different waveform.

2. **The process of generating the waveform**

```
%The process of generating the waveform
for n = 1:harmonics
    if mod(n,2) %Only consider the odd number
        ft =V0*sin(n*2*pi*f0*t)*(1/n); %The equation of ft
        ft_final = ft_final + ft; %Add them together
    end
end
```

Figure 4: The process of generating the waveform

In the process of plotting the figure, a loop about n is set and the f(t) obtained each time will be added together to obtain the final f(t). In addition, according to the formula provided, this algorithm only considers the **odd number** case.

3. **The step for plotting the waveform**

The next part is to plot the waveform, the corresponding codes are as follows:

```
%Plot the final waveform
figure(1);
subplot(2,1,1)
plot(t,ft_final);
grid on
xlabel('Time(t)') %Time
ylabel('Amplitude(ft)')
title('Resulting waveform')
```

Figure 5: The steps for plotting the waveform

In the process of plotting the waveform, the abscissa is set to Time(t), and the ordinate is set to Amplitude(ft). According to the formula of f(t) provided, a_n is 0. Therefore, the value of c_n will be equal to the b_n.

4. **The resulting waveform**

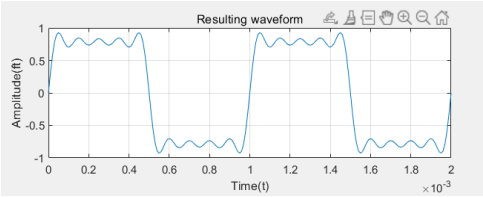


Figure 6: The resulting waveform

The specific description of the figure will be introduced in the next question.

- **Question A.2**

  Explain the features of the resulting waveform (peak-to-peak amplitude, symmetry, ripple, etc.) [5 marks]

  1. **Peak-to-peak amplitude**

     It can be seen from the figure that the maximum value of the waveform is 0.928V, and the minimum value is -0.928V in one period. Therefore, the final peak to peak voltage will be:

     $$Amplitude_{p-p} = V_{max} - V_{min} = 0.928 - (-0.928) = 1.856V \tag{9}$$

     The voltage is not very close to the "theoretical value" (If n approaches infinity). When V0 is assumed to be 1V and harmonic approaches infinity, the peak to peak amplitude of the theoretical square wave may be close to 1.56V (Simulated value when harmonic =10000). When only the first 10 harmonics are considered, there will be some differences because of Gibbs phenomena.

  2. **Symmetry**

     It can be found from the figure that the waveform has symmetry. When the assumed f0 is 1000 Hz, the waveform will be symmetrical about x=1. By increasing the time, it can be found that there are countless symmetry axes of the waveform and the waveform will be symmetric about x=n (n is integer).

  3. **Ripple**

     The series of ripples appear in the waveform of each period. And in each area where ripples appear, the amplitude of the ripple in the begin and final positions of the waveform will be the largest, and the amplitudes in the middle part will be basically similar and smaller. And in one period of the waveform, the range of ripple will be not continuous. In the middle of each period, the wave drops rapidly, and then ripples again. The rippled time of the wave occupies most of one period. When calculating the specific overshoot percentage of the synthesized waveform, the following formula may be needed:

     $$Percentage = \frac{V_{max} - V_{steady}}{V_{steady}} \tag{10}$$

     Through this method, the percentage overshoot of this waveform will be close to 18.1% when harmonic=10. If continue to increase the value of harmonic, the ripple range of this waveform will decrease.

4. **Period and the range of time**

   Since f0=1000Hz is set in the variable declaration and the period of this synthesised waveform is 1 second. In this experiment, the time of two periods will be selected (2 seconds).

- **Question A.3**

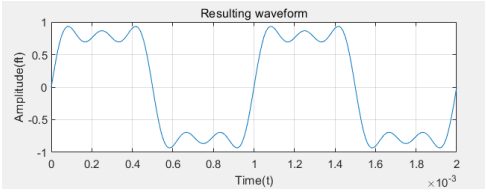
  How do you think the waveform would look if an unlimited number of harmonics was available (i.e. n goes to)? To support your answer, provide a couple of figures along with their associated code. [5 marks]

  If n goes ∞(unlimited number of harmonics), the waveform will become a square wave. Because the design of this part of the code takes into account the situation of changing the harmonic, therefore, people can directly choose a suitable value of harmonic to get different waveforms. The figures below are the obtained waveforms with several different harmonic values (5, 20, 100, 1000), and the above rules can also be summarized through these figures. When changing the harmonic, people only need to change the harmonic value in the following code.

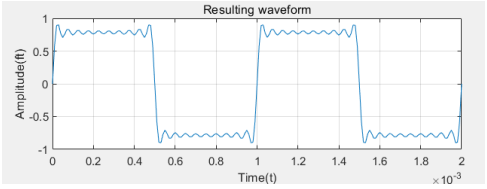
```
%Assume f0 = 1000Hz and V0=1V
f0 = 1000;
V0=1;

t = 0:0.00001:0.001; %Define the time variable
harmonics =10; %In part A, the harmonic will change
ft_final = 0; %Initialize the function
```

Figure 7: Change the value of the harmonic

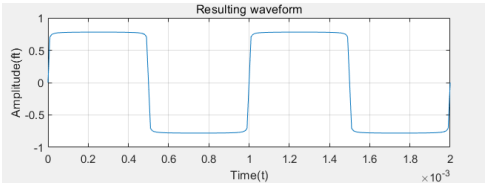
In this experiment, 4 different harmonics are set and the figures below are the waveforms corresponding to each harmonic.



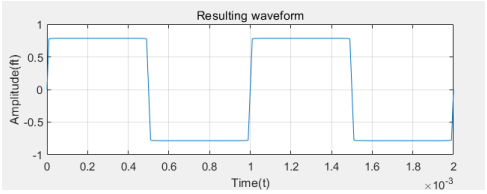
(a) The waveform for harmonic=5



(b) The waveform for harmonic=20



(c) The waveform for harmonic=100



(d) The waveform for harmonic=1000

Figure 8: The waveform for different harmonics

It can be seen from the figure that with the increase of harmonic, the rippled amplitude of the synthetic waveform becomes smaller and smaller. When harmonic=1000, the existence of ripple will be basically not observed. Therefore, it can be assumed that when harmonic approaches **infinity**, the waveform will approximately become a **square wave**.

- **Question A.4**

  Referring to Equation 3, what is the value of a_0? [5 marks]

  By checking the lab script, the equation 3 is:

$$a_0 = \frac{1}{T}\int_0^T f(t)dt \tag{11}$$

  Therefore, it can be calculated by this formula:

$$a_0 = \frac{1}{T}\int_0^T f(t)dt = \frac{1}{1}\int_0^1 sin(\omega*t)+\frac{1}{3}*sin(3*\omega*t)+\frac{1}{5}*sin(5*\omega*t)+\frac{1}{7}*sin(7*\omega*t)+\frac{1}{9}*sin(9*\omega*t)dt = 0 \tag{12}$$

  Where $V_0 = 1$,T=1

  The calculation process can also be simplified. According to the calculation formula provided, a_0 is actually the **average value of the amplitude** of the waveform, because the synthetic waveform is symmetric about the x-axis. Therefore, the average value is 0 which means a_0 will be equal to 0.

- **Question A.5**

  Find an expression (in terms of n) for a_n and b_n. [5 marks]

  According to the formula provided by the lab script, the following equations will be used [2]:

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n * cos(2*\pi*n*f_0*t) + \sum_{n=1}^{\infty} b_n * sin(2*\pi*n*f_0*t) \tag{13}$$

  Therefore, a_n is the coefficient of the cosine term in the equation of f(t) through observing the formula. Since there is no cosine term in the f(t) equation provided by part A, a_n **will only be 0** .

  Similarly, considering that b_n is the coefficient of the sine term in the formula f(t). According to the lab script, b1=1, b3=3, b5=5..., which means that only the coefficient of odd number is considered in this formula. Therefore, b_n will be:

$$b_n = 1/n \tag{14}$$

  Where n is odd number (1,3,5,7...)

- **Question A.6**

  Plot the spectrum (frequency domain view) of f(t) using $c_n = \sqrt{a_n^2 + b_n^2}$. Provide the figure and the Matlab code used to obtain it (you can use the stem function from Matlab to plot the frequency components). [5 marks]

  As the above calculation, a_n will be equal to 0 which means that $c_n = b_n$. With this condition, a loop about n will also be used. The part "define the variables and the initialization function" will not be changed, only the algorithm part will change. The specific code can be viewed below. The whole code can be viewed in the appendix.

```
%Plot the spectrum(frequency domain view)
for n = 1:harmonics
    if mod(n, 2) %Only consider the odd number
        bn=1/n; %Because an=0, cn will be equal to bn
        X=n;
    end

%Plot discrete sequence data
subplot(2, 1, 2)
stem(X, bn)
title('Spectrum(frequency domain view)')
xlabel('f(w)') %Frequency
ylabel('Magnitude')
hold on
end
```

Figure 9: The specific algorithm part

It can be found from the codes that the stem() function is used to plot discrete sequence with only considering odd numbers. The function of stem() is for plotting discrete sequence data because $b_n = 1/n$ and n is odd number. Therefore, mod(n, 2) will be also used.

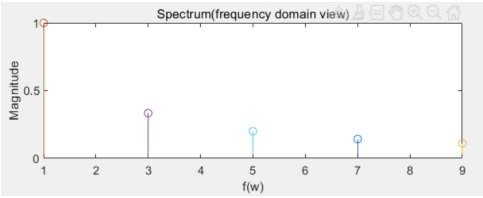


Figure 10: The spectrum (frequency domain view)

From the perspective of theoretical analysis, since in part A, the value of harmonic is 1, 3, 5, 7, 9. Therefore, there will be five discrete points corresponding to each frequency. By observing the figure, the discrete data only appears on the straight line x=odd number. And the **magnitude keeps decreasing as the frequency increases**. The magnitudes of the first few magnitudes are (1,1/3,1/5,1/7...), which is also the value of c_n.

2.2 Part B (20 marks)

2.2.1 Part B requirements

In this task, sawtooth waveform will be used for simulation. The expression of a typical sawtooth waveform is as follows:

$$f(t) = V - \frac{2 * V}{t}, 0 \leq t < T \tag{15}$$

According to this equation, the original and synthesized sawtooth waveforms in different harmonics(5 and 10) will be plotted.

2.2.2 Answer the questions

In this section, each question will be answered in the order of the questions provided in the template. The whole codes will be provided in the appendix according to different parts.

- **Question B.1 Write f(t) in Fourier synthesis form, i.e. as in Equation 2. [4 marks]**

According to the lab script, the equation 2 is:

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n * cos(2 * \pi * n * f_0 * t) + \sum_{n=1}^{\infty} b_n * sin(2 * \pi * n * f_0 * t) \tag{16}$$

In addition, according to the lab script, a_0 and a_n will be equal to 0 and $bn = 2 * V/\pi * n$. Therefore, the Fourier synthesis form of f(t) will be :

$$f(t) = \sum_{n=1}^{\infty} \frac{2 * V}{n * \pi} * sin(2 * \pi * f_0 * n * t) \tag{17}$$

If V=1V is assumed, the equation will be:

$$f(t) = \sum_{n=1}^{\infty} \frac{2}{n * \pi} * sin(2 * \pi * f_0 * n * t) \tag{18}$$

In the next questions, V will be assumed to 1V.

- **Question B.2 Calculate the first 10 sine wave coefficients (i.e. b1, b2, ... , b10). [4 marks]**

  According to the following formula, the value of b_n can be calculated when n is 1, 2, 3...10.

$$b_n = \frac{2 * V}{n * \pi} \tag{19}$$

  where n is 1,2,3, ... 10

  Therefore, the final results will be (Assuming the value of V is 1V.):

Table 1: The calculated values of b_n

b1	b2	b3	b4	b5	b6	b7	b8	b9	b10
$\frac{2*V}{\pi}$	$\frac{2*V}{2*\pi}$	$\frac{2*V}{3*\pi}$	$\frac{2*V}{4*\pi}$	$\frac{2*V}{5*\pi}$	$\frac{2*V}{6*\pi}$	$\frac{2*V}{7*\pi}$	$\frac{2*V}{8*\pi}$	$\frac{2*V}{9*\pi}$	$\frac{2*V}{10*\pi}$
0.637	0.318	0.212	0.159	0.127	0.106	0.091	0.080	0.071	0.064

- **Question B.3 Synthesise the first 10 harmonics of this waveform and plot the result (provide your Matlab code as well). [4 marks]**

  According to the lab script, the standard sawtooth waveform is as follows:

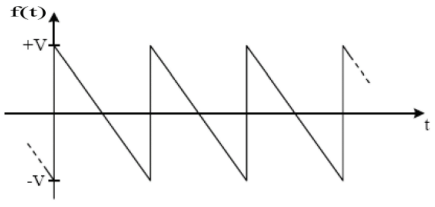


Figure 11: Sawtooth waveform(taken from [2])

The steps to plot sawtooth waveform with matlab are as follows (the whole codes can be viewed in the appendix)

1. **Define the variables and initialize the f(t) function**

```
%Define the period and V
T = 1;
V = 1;
w0 = 2*pi; % w=2*pi/T

%Define an and bn
n_max = 50; %Define the max n
n = 1:n_max;
an = 0; a0 = 0;
bn = 2*V./(n*pi);

%Define the time variabler
t = 0:1/50:3; %From 0s to 3s

%Initialize waveform f(t)
ft = zeros(size(t));
```

Figure 12: Define the variables and initialize the f(t) function

As can be found from the above, the period and V are assumed as 1s and 1V. The formulas of a_n and b_n are also defined with time variable. And the zeros() function is used to initialize the f(t) function.

2. **Plot the first 10 harmonics of this waveform**

```
%The first 10 harmonics of this waveform
n = 1:10;
for i = 1:length(t)

ft(i) = sum(bn(n).*sin(n*w0*t(i)));

end

%Plot the first 10 harmonics of this waveform
figure(1)
subplot(2,2,1);
plot(t,ft);
title(['The first 10 harmonics of this waveform']);
xlabel('Time(t)');
ylabel('Amplitude(ft)');
grid on;
```

Figure 13: The codes for plotting the waveform

Similar to other parts, the corresponding codes also use a **loop** about n. The specific synthesised waveform can be seen below:

3. **The resulting waveform when harmonic is 10**

It can be seen from the figure that when the harmonic is 10, the waveform will appear ripple phenomenon in most of the time in each period. Moreover, the shape of the waveform is close to the triangular wave. Although the rippled range of the waveform is not very obvious, it may be still a bit different from a straight line.

- **Question B.4**

Plot in the same figure the original and synthesised sawtooth waveforms (provide your Matlab code). Compare the resulting waveform with what you expected to see and discuss the results. [4 marks]

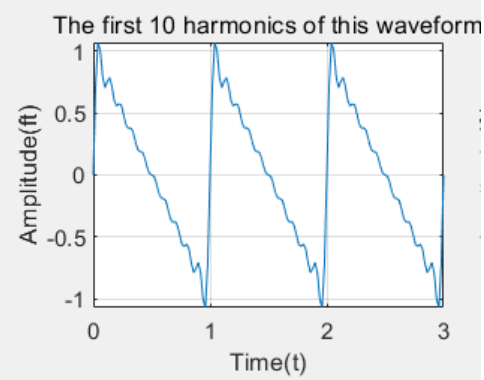


Figure 14: The resulting waveform when harmonic is 10

1. **Plot the original waveform**

```
%Plot the original signal
x = sawtooth(2*pi*t, 0);   %The original waveform
subplot(2, 2, 2);
plot(t, x);
xlabel('Time(t)');
ylabel('Amplitude(ft)');
grid on;
title('Original signal');
```

Figure 15: Plot the original waveform

In this part, sawtooth(t,xmax) is used to generate a modified triangle wave with the maximum location at each period controlled by nmax [3].

2. **The synthesised and original waveforms**

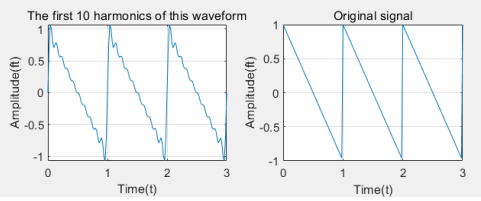


Figure 16: The synthesised and original waveforms

From the original figure, it can be found that the shape of the original waveform is exactly the same as the provided waveform in the lab script (considering the case of V=1V), so the original waveform obtained from the simulation is **in line with** expectations. Moreover, the shape of two waveforms are both close to the triangular waves. In addition, it can be found from the comparison that the original waveform does not appear ripple and all the lines are straight lines. But apart from this, the shapes of the two waveforms are very similar. In the following question , the effect of decreasing the harmonic value will be discussed.

- **Question B.5**

  If the number of harmonics is reduced to 5, comment on the changes that will be observed practically. [4 marks]

Because the part of initializing variables is the same and this part of the codes can be viewed in the previous question, only the codes and result of the specific algorithm part will be provided.

1. **Plot the first 5 harmonics of this waveform**

```
%The first 5 harmonics of this waveform
n = 1:5;
for i = 1:length(t)

ft(i) = sum(bn(n).*sin(n*w0*t(i)));

end

%Plot the first 5 harmonics of this waveform
subplot(2, 2, 3);
plot(t, ft);
title(['The first 5 harmonics of this waveform']);
xlabel('Time(t)');
ylabel('Amplitude(ft)');
grid on;
```

Figure 17: Plot the first 5 harmonics of this waveform

2. **The resulting waveform for harmonic=5**
   Similar to the method in part A, a loop is also used to plot the first 5 harmonics of this waveform. Note that **use ".*" instead of "*"** when writing the code.

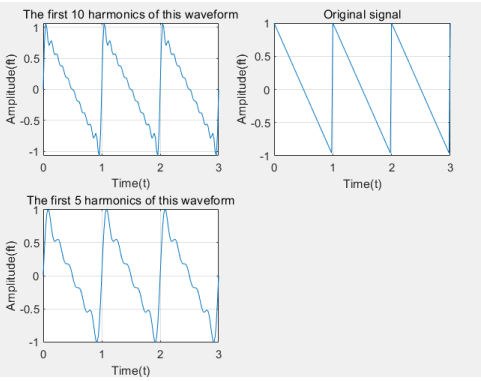


Figure 18: Plot the first 5 harmonics of this waveform

Through comparison, it can be found that when harmonic=5, the phenomenon of ripple is more obvious than that of harmonic=10 and the amplitude of ripple is bigger. In addition, the three figures are very similar in general shape. Since increasing the value of harmonic can reduce the amplitude of ripple phenomena and the shape will be more similar to triangular wave. Therefore, the rule **"when the value of harmonic tends to infinity, the resulting waveform figure will be very close to the original waveform"**can be summarized. The conclusion obtained is very similar to the conclusion in part A.

2.3 Part C (15 marks)

Since there are 4 different f(t) to be drawn and commented in this part, it will be divided into four individual parts corresponding to question a b c d.

2.3.1 Question a

In this question, the following formula about f(t) need to be drawn:

$$f_1(t) = sin\omega_0 t - \frac{1}{9}sin3\omega_0 t + \frac{1}{25}sin5\omega_0 t - \frac{1}{49}sin7\omega_0 t + ... \tag{20}$$

Therefore, the code provided and the results obtained are as follows:

1. **Define the variables and initialize the function**

```
%Assume w0 = 2*pi, then T=1
w0 = 2*pi;

t = 0:0.001:3; %Time variable
harmonics =100; %In this part, the harmonics is not fixed
ft_final = 0; %Initialize the function
```

Figure 19: Define the variables and initialize the function

As can be found from the above code, ω_0 is assumed as $2 * \pi$. And the final f(t) will be initialized to 0.

2. **The algorithm of synthesised waveform**

```
%The process of generating the waveform
for n = 1:harmonics
    if mod(n,2)%Only consider the odd number
        ft = (-1)^((n-1)/2)*sin(n*w0*t)/n^2; %The equation of ft
        ft_final = ft_final + ft; %Add them together
    end
end
```

Figure 20: The algorithm of synthesised waveform

The algorithm part also uses a loop about n. Similar to the previous part, the algorithm only considers odd numbers. Therefore, the mod() function is used here.

3. **Plot the waveform**

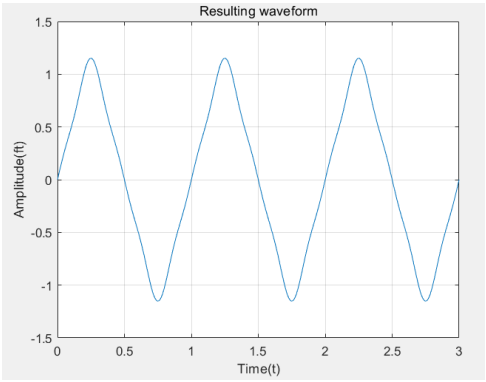
```
%plot the resulting waveform
figure;
plot(t,ft_final);
grid on
xlabel('Time(t)')
ylabel('Amplitude(ft)')
title('Resulting waveform')
```

Figure 21: The codes for plotting the waveform

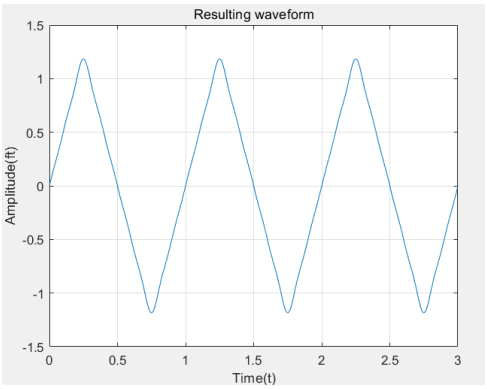
The method of plotting this synthesised waveform is the same as the previous part

4. **Results and comments for this waveform**

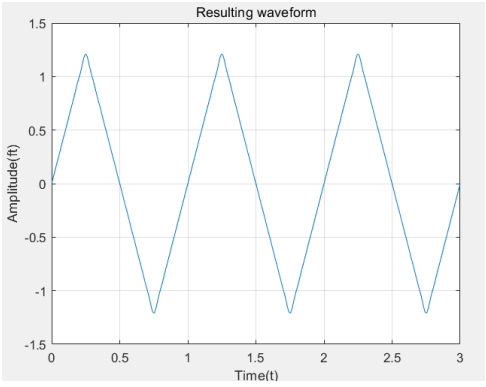
Since the specific harmonic value is not clearly provided in the question, four different waveforms will be provided in the result according to different harmonic value (5,10,20,100)



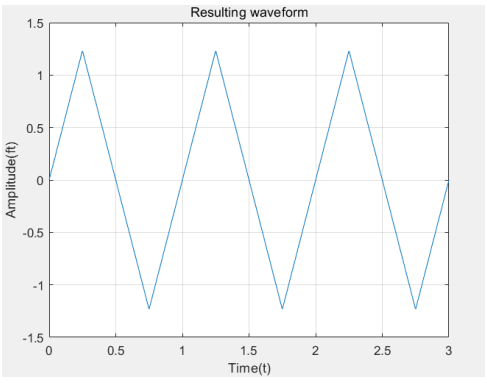
(a) The waveform for harmonic=5



(b) The waveform for harmonic=10



(c) The waveform for harmonic=20



(d) The waveform for harmonic=100

Figure 22: The waveforms for different harmonics

If $w = 2\pi$ is assumed, the period of all waveforms is 1 second. Each set of waveforms has a "wave" shape and the amplitude of the initial position and the end position in the same period is the same, both are 0. The waveform first rises to the maximum point in the first quarter of the period (maximum value approximate to 1.2V). Then the curve begins

to fall from first quarter of the period to the three quarters of the period, reaching the lowest point (the lowest point is about -1.2V). Then the waveform rises again to 0V.

The function in question a is periodic. Additionally, the common points of all harmonic waveforms are that the shapes of all the waveforms are very similar and the rise and fall times in one period are basically the same, which means there are no phase differences. The different points are that as the harmonic increases, each segment of the resulting waveform gradually becomes straight, and the range of rippled phenomena becomes smaller. Therefore, when the harmonic is large enough, the waveform will become a perfect "triangular" shape, just like the waveform with harmonic=100.

2.3.2 Question b

In this question, the following formula about f(t) need to be drawn:

$$f_2(t) = 0.1(cos\omega_0 t + cos2\omega_0 t + cos3\omega_0 t + cos4\omega_0 t + ...) \tag{21}$$

Therefore, the codes provided and the results obtained are as follows. Since the code for initializing and plotting the figure in this part is the same as the above, only the algorithm codes will be discussed here. The whole code can be viewed in the appendix.

1. **Synthesise the (b) waveform**

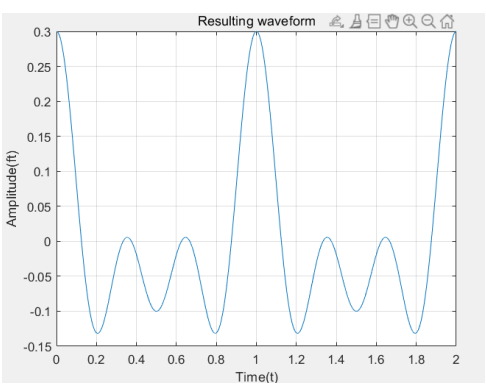
```
%The process of generating the waveform
for n = 1:harmonics
    if mod(n,2)%0nly consider the odd number
        f1t= 0.1*cos((n+1)/2*w0*t);%The equation of ft
        ft_final= ft_final + f1t;%Add them together
    end
end
```

Figure 23: The codes for plotting the waveform

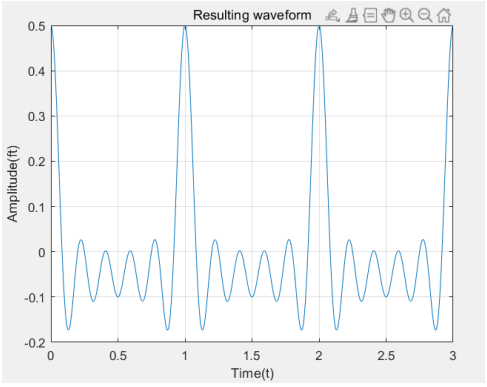
The algorithm part also uses a loop about n. Similar to the previous part, the algorithm only considers odd numbers. Therefore, the mod() function is used here.

2. **Results and comments for this waveform**

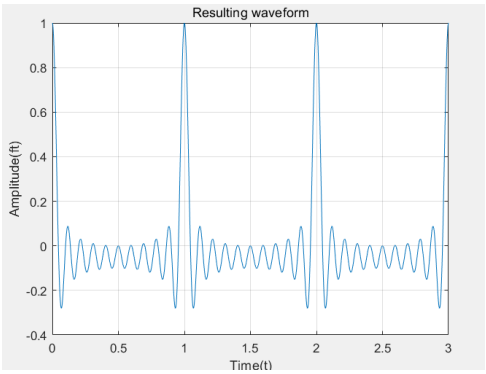
Since the specific harmonic value is not clearly provided in the question, four different waveforms will be provided in the result according to different harmonic value (5,10,20,100)



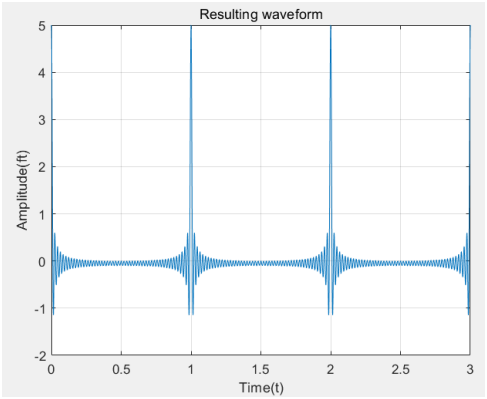
(a) The waveform for harmonic=5



(b) The waveform for harmonic=10



(c) The waveform for harmonic=20



(d) The waveform for harmonic=100

Figure 24: The waveforms for different harmonics

The function in question b is periodic. If $w = 2\pi$ is assumed, the period of all waveforms is 1 second. In one period, all waveforms ripple most of the time. And the amplitude of the initial position and the end position in one period are the same. All the waveform shapes are very similar and the rise and fall times are basically the same, which means that there are no phase differences. The waveform starts to drop from the point of the initial position, and it rises and falls (ripple) at the position of amplitude=0V at most of the time in the middle. But with the increase of harmonic, the rippled amplitude in the middle part will become smaller. When the harmonic is large enough, the overshoot of the waveform will be very small. This is also Gibbs effects, as the increase of harmonic, the ripple of the waveform will gradually decrease and it will finally reach a steady state and the overshoot and ripple will be hard to observe [4].

2.3.3 Question c

In this question, the following formula f(t) need to be plotted:

$$f_3(t) = sin\omega_0 t - \frac{4}{3\pi}cos2\omega_0 t - \frac{4}{15\pi}cos4\omega_0 t - \frac{4}{35\pi}cos6\omega_0 t - \frac{4}{63}cos8\omega_0 t - \frac{4}{99\pi}cos10\omega_0 t \quad (22)$$

Since the harmonic is fixed this time, the waveform of f(t) above will be plotted directly. Therefore, the codes and the results obtained are as follows:

1. **Define the variables**

```
%Assume w0 = 2*pi, then T=1
w0 = 2*pi;

t = 0:0.001:1; %Time variable
harmonics =10; %In this part, the harmonics is not fixed
ft_final = 0; %Initialize the function
```

Figure 25: The codes for defining the variables

Only ω_0 and time variable are defined.

2. **Plot the waveform**

```
%The equation of ft
ft = sin(w0*t)-(4/(3*pi))*cos(2*w0*t)-(4/(15*pi))*cos(4*w0*t)-...
    (4/(35*pi))*cos(6*w0*t)-(4/(63*pi))*cos(8*w0*t)-(4/(99*pi))*cos(10*w0*t);
```

Figure 26: The codes for plotting the waveform

3. **Results and comments for this waveform**

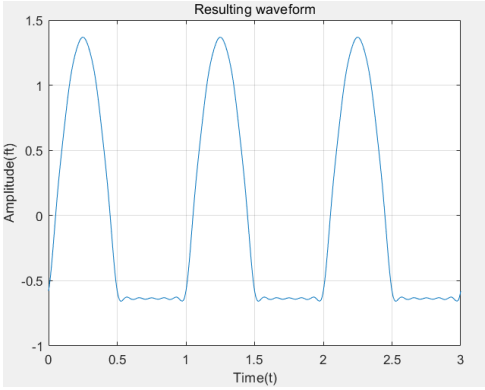


Figure 27: The resulting waveform

This function f(t) is **periodic** all the time. If $w = 2\pi$ is assumed, the period of this waveform will be 1 second. Since the number of sine and cosine items here is fixed, the specific figure of f(t) can be plotted directly.

It can be seen from the figure that the wavefrom seems like be "clipped" from half of the period to the final time in one cycle. Additionally, the shape of the waveform shows the shape of "arc" in the first half of the period, but Gibbs phenomenon appears in the second half of the period with rippling and overshooting. Although compared with the waveform obtained in the previous questions, the amplitude of rippled phenomena is not very large. Similarly, the amplitude of the waveform at the initial position and the final position is the same. The maximum amplitude in one period is about 1.36V and the minimum amplitude is about -0.65V.

2.3.4 Question d

In this question, the following formula f(t) need to be plotted(Switched off the fundamental component):

$$f_3(t) = -\frac{4}{3\pi}cos2\omega_0 t - \frac{4}{15\pi}cos4\omega_0 t - \frac{4}{35\pi}cos6\omega_0 t - \frac{4}{63}cos8\omega_0 t - \frac{4}{99\pi}cos10\omega_0 t \qquad (23)$$

Since the harmonic is also fixed this time, the waveform of f(t) above will be plotted directly. Therefore, the codes and the results obtained are as follows. The codes for initialization and plotting are the same as question c.

1. **Plot the waveform**

```
%The equation of ft
ft = -(4/(3*pi))*cos(2*w0.*t)-(4/(15*pi))*cos(4*w0.*t)-...
    (4/(35*pi))*cos(6*w0.*t)-(4/(63*pi))*cos(8*w0.*t)-(4/(99*pi))*cos(10*w0.*t);
```

Figure 28: The codes for plotting the waveform

In the question d, f(t) is composed of all cosine items because the first sine item is switched off.

2. **Results and comments for this waveform**

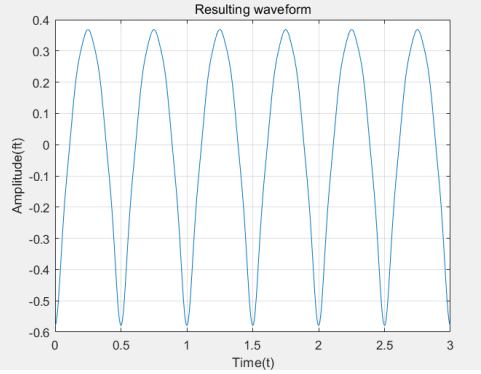


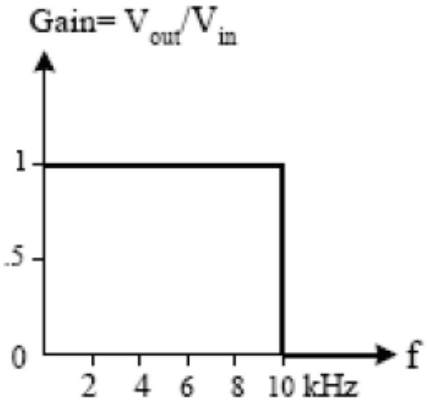
Figure 29: The waveform for different harmonics

The function in question d is periodic. If $w = 2\pi$ is assumed, the period of this waveform is 1 second. The maximum amplitude of this waveform in one period is about 0.37V and the minimum amplitude is about -0.58V. Compared with question c which did not switch off sine, there was **no Gibbs phenomena** in the second half of the period with no ripple. In the 0-1/4 period, the waveform rises, in 1/4-1/2 period, the waveform drops from the highest point to the lowest point. Moreover, in 1/2-3/4 period, the waveform rises again to the highest point and in 3/4-1 period, the waveform drops from the maximum point to the lowest point again.

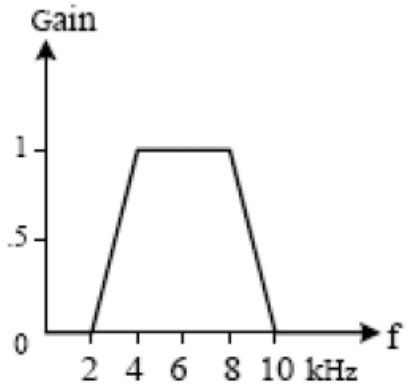
2.4 Part D (15 marks)

2.4.1 Part D requirements

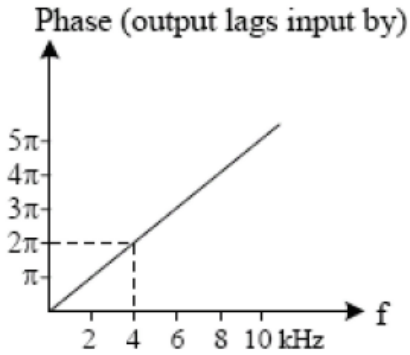
In this question, two pass filter and phase response are both considered. Therefore, there are four parts of code in Part D: Plotting the original function, Plotting A low-pass filter with two different phase responses and A band-pass filter with one phase responses. Among them, the schematic diagrams of pass-filter and two phase responses are as follows:



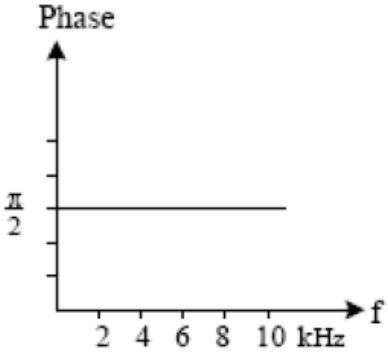
(a) Low-pass filter gain response



(b) Band-pass filter gain response



(c) Example for one phase response



(d) Example for another phase response

Figure 30: Four kinds of gain and phase responses

2.4.2 Answer the questions

1. **Question D.1 Synthesise and plot this square wave (provide your Matlab code as well). Calculate the percentage overshoot of the synthesised waveform (compared with the ideal waveform) at the discontinuity. How does this compare with the expected limit of 17.9%? [2 marks]**

   First, the codes and the resulting waveform are as follows

   (a) **Define the variables and initialize the function**
   First, f_0 and time variable are defined. In addition, f(t) is also initialized in this part. Because the harmonic is not specified to a accurate value. Therefore, the value of harmonic needs to be changed during simulation.

```
%assume f0 = 1000Hz
f0 = 1000;
t = 0:0.00001:0.005; %Time variable
harmonics = 20; %In this part, the harmonics is not fixed
ft_final = 0; %Initialize the function
```

Figure 31: Define the variables and initialize the function

(b) **The algorithm of synthesised waveform**

```
for n = 1:harmonics
    if mod(n,2)%Only consider the odd number
        ft =(3/pi)*sin(n*2*pi*f0*t)*(1/n);%The equation of ft
        ft_final = ft_final + ft;%Add them together
    end
end
os=overshoot(ft_final);%The percentage overshoot of the synthesised waveform
```

Figure 32: The algorithm of synthesised waveform

As in the previous sections, a loop about n is also used. Since only the odd number is considered, the mod() function is also used.

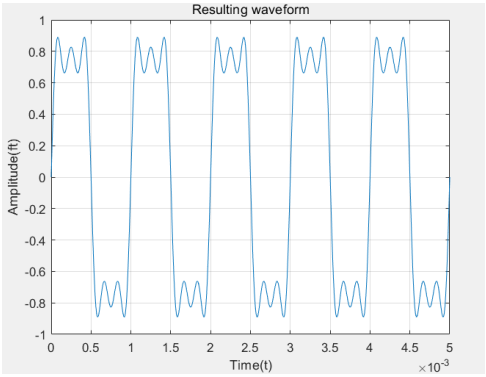
(c) **Plot the waveform**

```
%plot the resulting waveform
figure;
plot(t, ft_final);
grid on
xlabel('Time(t)')
ylabel('Amplitude(ft)')
title('Resulting waveform')
```

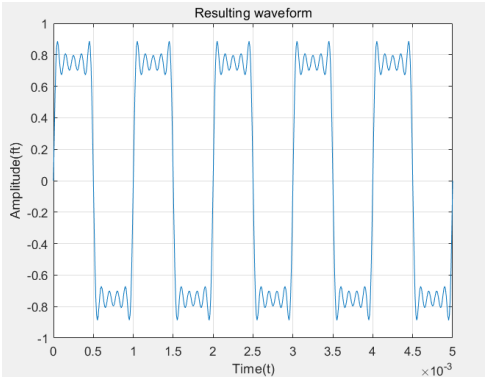
Figure 33: The codes for plotting the waveform

The above codes are used to plot the waveform.

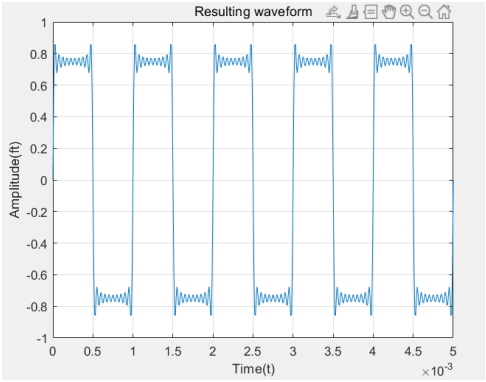
(d) **The resulting waveforms when harmonic is 5,10,20,100**



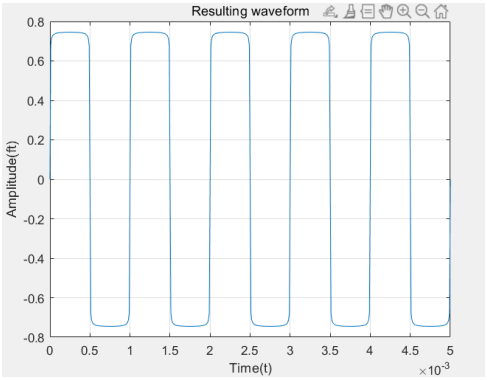
(a) The waveform for harmonic=5



(b) The waveform for harmonic=10



(c) The waveform for harmonic=20



(d) The waveform for harmonic=100

Figure 34: The waveforms for different harmonics

According to the above waveform analysis, as the value of harmonic increases, the shape of the waveform will be more similar to the square wave.

(e) **Calculate the percentage overshoot and compare this value to the expected limit**

According to the formula:

$$Percentage = \frac{V_{max} - V_{steady}}{V_{steady}} \tag{24}$$

Where V_{steady} is the steady voltage, which is hard to know directly. Therefore, the voltage when n goes to ∞ will be chosen because it is approximate to the steady voltage in each waveform. And the V_{max} is the maximum value of the waveform of one period

The specific results are as follows:

Table 2: Each percentage of overshoot

	Harmonic=5	Harmonic=10	Harmonic=100	Harmonic=1000
Vmax or steady value(V)	0.8913	0.8867	0.8843	0.8842
Harmonic=10000(Infinity)	0.75	0.75	0.75	0.75
Percentage(%)	18.84	18.23	17.91	17.89

Since the expected limit is 17.9%, when harmonic=5, the percentage obtained will be greater than 17.9% indicating that the overshoot value is relatively large, which means that the waveform will ripple more obviously when the harmonic is relatively small. As the harmonic increases, the percentage of overshoot will gradually tend to the expected limit of 17.9%. This is in line with expectations. Because as the harmonic gradually increases, the ripple will become smaller, which causes the value of overshoot to be smaller. From the above results, it can be found that when Harmonic=1000, the calculated percentage is very close to the expected limit. The following is the specific explanation consulted on Matlab [3]:

For a positive-going (positive-polarity) pulse, overshoot expressed as the percentage will be :

$$Percentage = 100 * \frac{O - S_2}{S_2 - S_1} \tag{25}$$

Where O is the max deviation greater than the high level and S2 is high level,S1 is the low level.The specific explanation can be seen below:

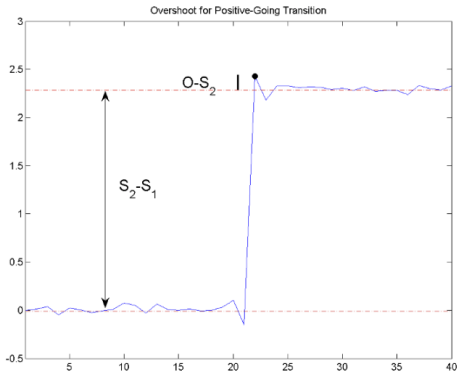


Figure 35: The figure for the calculation of overshoot(taken from [3])

The red dashed line indicates the estimated state level. The double-sided black arrow means the difference between the high and low levels. The black solid line represents the difference between the overshoot value and the high level(S1) [3].

2. **Question D.2 What is the name of this overshoot? Explain it. [2 marks]**

   Gibbs Phenomenon

   Gibbs phenomenon reflects the difficulty in approximating a discontinuous function by a finite series of continuous sine and cosine waves [4]. Although every partial sum of the

Fourier series overshoots the function it is approximating, the limit of the partial sums does not [4]. Moreover, the value of x where the maximum overshoot is achieved moves closer and closer to the discontinuity as the number of terms summed increases.

From the perspective of signal processing, Gibbs phenomenon is the step response of a low-pass filter, but the overall integral does not change under filtering. After checking some information on the internet, the **sine integral cause overshoot**, which related to the Gibbs phenomenon [4]. The sine integral can be considered as the convolution of the sinc function with the step function, which is related to truncating the Fourier series.

3. **Question D.3 (a) Give the Fourier series in each case for the resulting waveform if the above square wave is used as an input for:**

   (a) **Fourier series in question a**
   Since only odd numbers are considered in question a, the Fourier series will be:

$$f(t) = \sum_{1}^{10} \frac{3}{\pi} * sin(2\pi * f_0 * n * t - \frac{n * \pi}{2}), n = 1, 3, 5, 7, 9 \tag{26}$$

   Since only the odd number is considered, the value of n is 1, 3, 5, 7, 9, or it can be changed to the following form:

$$f(t) = \frac{3}{\pi}(sin(2\pi * f_0 * t - \frac{\pi}{2}) + \frac{1}{3}sin(2\pi * f_0 * 3t - \frac{3\pi}{2}) + \frac{1}{5}sin(2\pi * f_0 * 5t - \frac{5\pi}{2}) + \frac{1}{7}sin(2\pi * f_0 * 7t - \frac{7\pi}{2}) + \frac{1}{9}sin(2\pi * f_0 * 9t - \frac{9\pi}{2})$$

Figure 36: The new Fourier series

   After simplification,

$$f(t) = -\frac{3}{\pi}(cos(2\pi * f_0 * t) + \frac{1}{3}cos(2\pi * f_0 * 3t) + \frac{1}{5}cos(2\pi * f_0 * 5t) + \frac{1}{7}cos(2\pi * f_0 * 7t) + \frac{1}{9}cos(2\pi * f_0 * 9t))$$

Figure 37: The simplified Fourier series

   It can be found that all sine items have been transformed into cosine items.

   (b) **Fourier series in question b**

$$f(t) = \sum_{1}^{10} \frac{3}{\pi} * sin(2\pi * f_0 * n * t - \frac{\pi}{2}), n = 1, 3, 5, 7, 9 \tag{27}$$

   Since only the odd number is considered, the value of n is 1, 3, 5, 7, 9, or it can be changed to the following form:

$$f(t) = \frac{3}{\pi}(sin(2\pi * f_0 * t - \frac{\pi}{2}) + \frac{1}{3}sin(2\pi * f_0 * 3t - \frac{\pi}{2}) + \frac{1}{5}sin(2\pi * f_0 * 5t - \frac{\pi}{2}) + \frac{1}{7}sin(2\pi * f_0 * 7t - \frac{\pi}{2}) + \frac{1}{9}sin(2\pi * f_0 * 9t - \frac{\pi}{2}))$$

Figure 38: The final Fourier series

(c) **Fourier series in question c**

In this question, it is necessary to consider that the gain response is not 1 all the time. Through observing the lab script, the gain for 3KHz and 8KHz is approximate 0.5, and the gain for 5KHz and 7KHz is 1 (Only consider the odd harmonics). Therefore, the specific Fourier series is:

$$f(t) = \frac{3}{\pi}(0.5 * \frac{1}{3}sin(2\pi * f_0 * 3t - \frac{3\pi}{2}) + \frac{1}{5}sin(2\pi * f_0 * 5t - \frac{5\pi}{2}) + \frac{1}{7}sin(2\pi * f_0 * 7t - \frac{7\pi}{2}) + 0.5 * \frac{1}{9}sin(2\pi * f_0 * 9t - \frac{9\pi}{2}))$$

Figure 39: The final Fourier series

4. **Question D.4 Synthesise the above waveforms and draw the obtained waveforms for filters (a), (b) and (c), providing the Matlab code as well. [3 marks]**

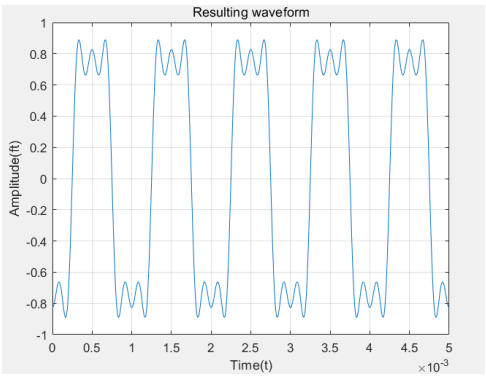
In this part, low-pass filter gain and the first phase response are used, so the codes and the specific part of the waveform are as follows. Since the initialization is the same as above, this part only shows the codes of the algorithm part:

(a) **The algorithm of synthesised waveform**

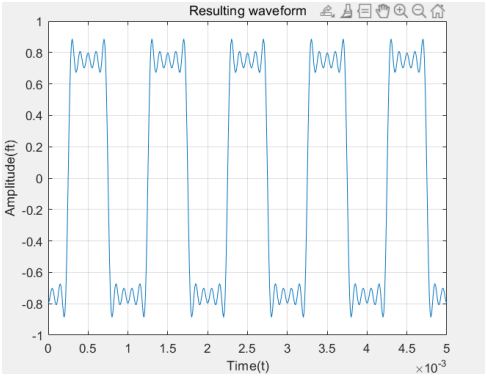
```
for n = 1:harmonics
    if(harmonics<=10) %Low-pass filter
        if mod(n,2) %Only consider the odd number
         ft =(3/pi)*sin(n*2*pi*f0*t-(n*pi/2))*(1/n);%The equation of ft
         ft_final = ft_final + ft;%Add them together

         %plot the resulting waveform
         figure(1);
         plot(t,ft_final);
         grid on
         xlabel('Time(t)')
         ylabel('Amplitude(ft)')
         title('Resulting waveform')
        end

    else
       %The first 10 harmonics of this waveform
       ft_final = (3/pi)*(sin(2*pi*f0*t-pi/2)+1/3*sin(2*pi*3*f0*t-(3*pi/2))+...
           1/5*sin(2*pi*5*f0*t-(5*pi/2))+1/7*sin(2*pi*7*f0*t-(7*pi/2))+1/9*sin(2*pi*9*f0*t-(9*pi/2)));

       %plot the resulting waveform
         figure(1);
         plot(t,ft_final);
         grid on
         xlabel('Time(t)')
         ylabel('Amplitude(ft)')
         title('Resulting waveform')
   end
end
```

Figure 40: The algorithm of synthesised waveform

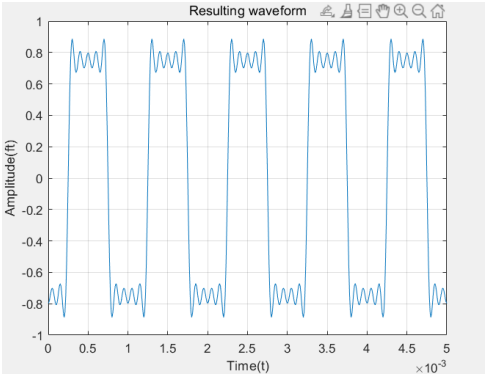
(b) **The resulting waveforms when harmonic is 5,10,20,100**



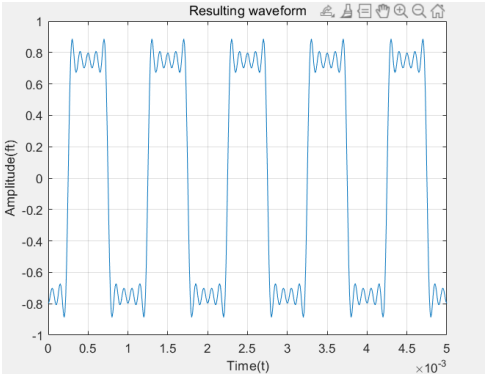
(a) The waveform for harmonic=5



(b) The waveform for harmonic=10



(c) The waveform for harmonic=20



(d) The waveform for harmonic=100

Figure 41: The waveforms for different harmonics

When harmonic is greater than 10, all the waveforms are the same. This is because the frequencies exceeding 10KHz will be filtered. Additionally, this waveform has a phase difference comparing with the original.

5. **Question D.3 (b) A low-pass filter with gain and phase responses as given in Figures 8 and 11 respectively. [1 mark]**

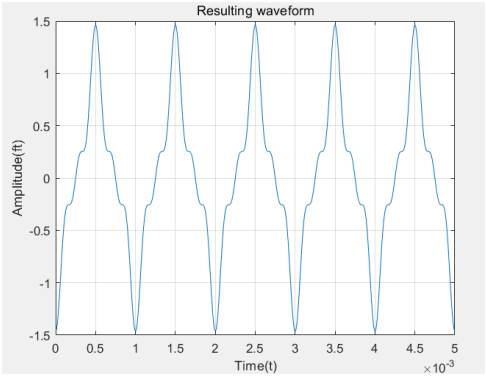
   Low-pass filter gain and the second phase response (lagging $-\frac{\pi}{2}$) are both used. The code and the specific waveform obtained are as follows. Since the initialization is the same as the above, this part only shows the code of the algorithm part

(a) **The algorithm of synthesised waveform**

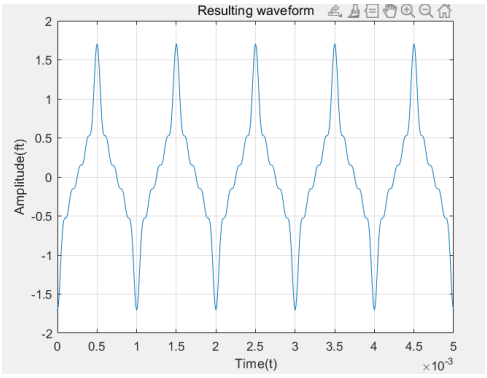
```
for n = 1:harmonics
    if(harmonics<=10) %Low-pass filter
        if mod(n,2) %Only consider the odd number
         ft =(3/pi)*sin(n*2*pi*f0*t-(pi/2))*(1/n);%The equation of ft
         ft_final = ft_final + ft;%Add them together

         %plot the resulting waveform
         figure(1);
         plot(t,ft_final);
         grid on
         xlabel('Time(t)')
         ylabel('Amplitude(ft)')
         title('Resulting waveform')
        end

    else
       %The first 10 harmonics of this waveform
       ft_final = (3/pi)*(sin(2*pi*f0*t-pi/2)+1/3*sin(2*pi*3*f0*t-(pi/2))+1/5*sin(2*pi*5*f0*t-(pi/2))+...
       1/7*sin(2*pi*7*f0*t-(pi/2))+1/9*sin(2*pi*9*f0*t-(pi/2)));

       %plot the resulting waveform
         figure(1);
         plot(t,ft_final);
         grid on
         xlabel('Time(t)')
         ylabel('Amplitude(ft)')
         title('Resulting waveform')
   end
end
```

Figure 42: The algorithm of synthesised waveform

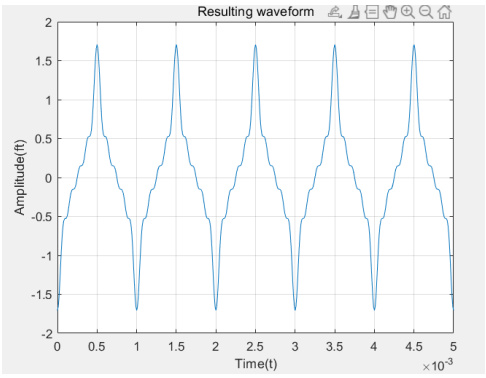
(b) **The resulting waveforms when harmonic is 5,10,20,100**



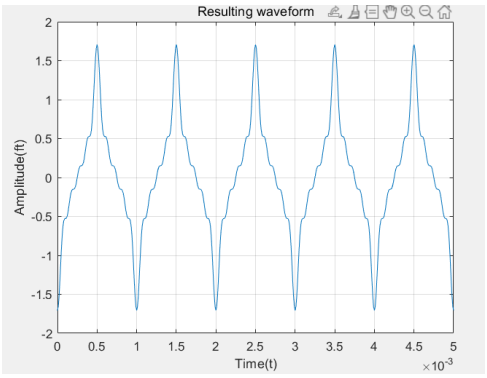
(a) The waveform for harmonic=5



(b) The waveform for harmonic=10



(c) The waveform for harmonic=20



(d) The waveform for harmonic=100

Figure 43: The waveforms for different harmonics

When harmonic is greater than 10, all the waveforms are the same. This is also

because the frequencies exceeding 10KHz will be filtered. The shape of this figure is also different comparing with the original waveform.

6. **Question D.3 (c)**

   A band-pass filter with gain and phase responses as given in Figures 9 and 10 respectively. [1 mark]

   Band-pass filter gain and the first phase response are used in this question. Because the gain response involved in this question is more complicated. Therefore, it is necessary to analyze the different gains of each segment under different frequencies.

   - When the frequency is 0-2KHz, gain response is 0 which means the output of this part is also 0.
   - When the frequency is 3-4KHz, the gain response of 3KHz is 0.5, and the gain response of 4KHz is 1.
   - When the frequency is 5-8KHz, all gain responses are 1.
   - When the frequency is 9-10KHz, the gain response of 9KHz is 0.5 and the gain response of 10KHz is 1.
   - When the frequency is greater than 10KHz, all gain responses are 0.

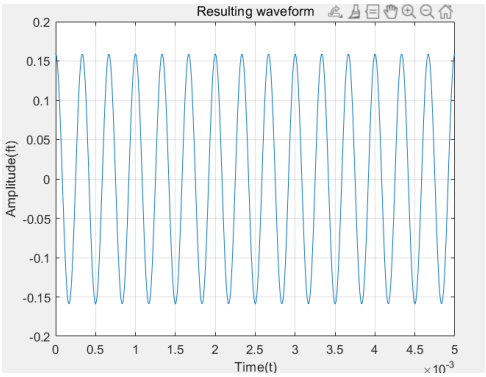
   Therefore, when designing the code, not only the phase differences must be considered, but also their corresponding coefficients (gain) must be considered as above. The code and the specific waveform obtained are as follows, because the initialization is the same as the above, so this part only shows the code of the algorithm part.

   (a) **The algorithm of synthesised waveform**

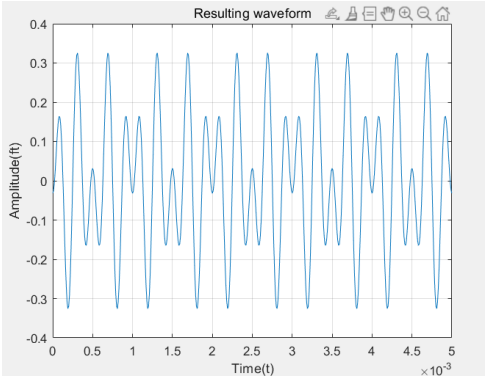
```
for n = 1:harmonics
    if(harmonics<=10)%Low-pass filter
      switch(harmonics)
        case{1, 2} %When the harmonics is 1 or 2
          ft_final=0;   %The equation of ft
        case{3, 4} %When the harmonics is 3 or 4
          ft_final=(3/pi)*0.5*1/3*sin(2*pi*3*f0*t-((pi/2)*3)); %The equation of ft
        case{5, 6} %When the harmonics is 5 or 6
          ft_final=(3/pi)*(0.5*1/3*sin(2*pi*3*f0*t-((pi/2)*3))+1/5*sin(2*pi*5*f0*t-((pi/2)*5))); %The equation of ft
        case{7, 8} %When the harmonics is 7 or 8
          ft_final=(3/pi)*(0.5*1/3*sin(2*pi*3*f0*t-((pi/2)*3))+1/5*sin(2*pi*5*f0*t-((pi/2)*5))+...
          1/7*sin(2*pi*7*f0*t-((pi/2)*7)));%The equation of ft
        case{9, 10} %When the harmonics is 9 or 10
          ft_final=(3/pi)*(0.5*1/3*sin(2*pi*3*f0*t-((pi/2)*3))+1/5*sin(2*pi*5*f0*t-((pi/2)*5))+...
              1/7*sin(2*pi*7*f0*t-((pi/2)*7))+0.5*1/9*sin(2*pi*9*f0*t-((pi/2)*9))); %The equation of ft
      end
    else
          ft_final=(3/pi)*(0.5*1/3*sin(2*pi*3*f0*t-((pi/2)*3))+1/5*sin(2*pi*5*f0*t-((pi/2)*5))+...
              1/7*sin(2*pi*7*f0*t-((pi/2)*7))+0.5*1/9*sin(2*pi*9*f0*t-((pi/2)*9))); %The equation of ft
    end
end
```

Figure 44: The algorithm of synthesised waveform

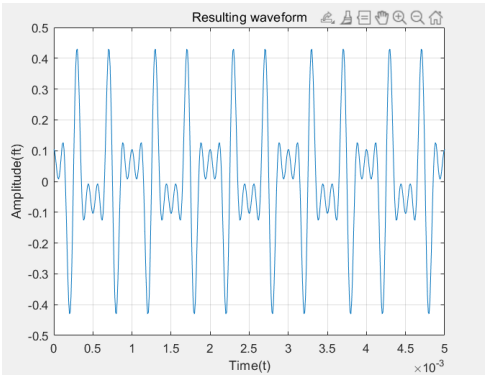
(b) **The resulting waveforms when harmonic is 3,5,7,9**



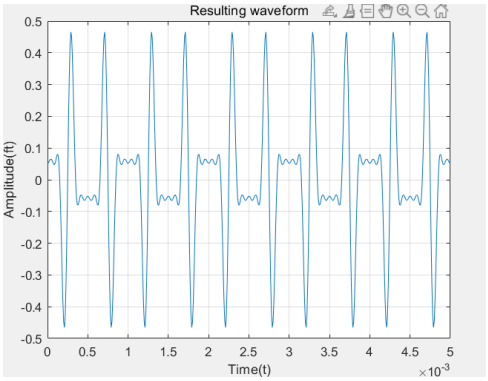
(a) The waveform for harmonic=3



(b) The waveform for harmonic=5



(c) The waveform for harmonic=7



(d) The waveform for harmonic=9

Figure 45: The waveforms for different harmonics

When harmonic is greater than 10, all waveforms are the same as the waveform when harmonic=9. This is because the low-pass filter will filter frequencies exceeding 10KHz. Since the gain response in each frequency range is not the same, the waveform will also show a difference when different harmonics are used. However, when harmonic is greater than 10, since the gain response is 0, all waveforms will be the same.

7. **Question D.5 What is the fundamental frequency of the output from filter (c)? Why? [5 marks]**

   Still 1KHz

   The reason is as follows: **Adding band-pass filter and increasing the phase shift will not affect the fundamental frequency.** Since the fundamental frequency in the experiment is 1KHz, the fundamental frequency of the final output signal will remain at 1KHz.

   From the other perspective, before defining fundamental frequency, the fundamental period need to be defined which is the smallest period over the function. And the fundamental frequency is defined as its reciprocal. The minimum period of this function has not changed, so the fundamental frequency will not change.

   In addition, A band-pass filter is an electronic device or circuit that allows signals between **two specific frequencies** to pass [5]. That means signals in a specific frequency band

are allowed to pass, while other frequency bands are shielded. Therefore, Band-pass filter only serves to filter the frequency, and does not affect the fundamental frequency.

For the phase filter response, it only changes the phase of the function, and has nothing to do with the frequency, so adding the phase filter response will not change the value of the fundamental frequency.

2.5 Part E (10 marks)

2.5.1 Part E requirements

In part E, three different frequencies of sounds and their chords will be simulated and played through the computer. In addition, the phase of the third harmonic will be reduced by $\frac{\pi}{2}$ and the simulation will be repeated. The equation of the original waveform is as follows:

$$V(t) = 10[sin(2\pi * f_0 t) + sin(2\pi * 2 * f_0 t) + sin(2\pi * 3 * f_0 t)] \tag{28}$$

2.5.2 Answer the questions

1. **Question E.1**

   For the wave in equation 15, listen to harmonics 1, 2 and 3 individually then as a chord. Plot the chord waveform as well. Provide the Matlab code used to listen to the harmonics/chord and plot the chord. [3 marks]

   The code and results are as follows:

   (a) **Define the variables**

```
%When harmonics is 1
fs = 44100; %Sampling frequency
duration = 3; %Duration for 3 seconds
t = 0 : 1/fs : duration; %Time variable

%Assume amplitude = 10 and f = 500
amplitude = 10;
f = 500;
```

Figure 46: Define the variables

According to the lab script, **fs,duration and time variable** need to defined individually. Additionally, the amplitude and frequency are assumed to 10V and 500Hz

(b) **The codes when harmonic=1,2,3 and chord**

```
wave1 = amplitude *sin(2*pi*f*t); %The equation of Vt when harmonics is 1
sound(wave1, fs); %Making a sound
pause(1*duration);%Pause

%Plot the waveform when harmonics is 1
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave1(1:find(t <= T_max/f, 1, 'last')); %The function of clipped waveform
figure
subplot(2, 2, 1);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The waveform when harmonics is 1');
hold on
```

(a) The waveform for first harmonic

```
wave2 = amplitude *(sin(2*pi*2*f*t)); %The equation of Vt when harmonics is 2
sound(wave2, fs); %Making a sound
pause(2*duration);%Pause

%Plot the waveform when harmonics is 2
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave2(1:find(t <= T_max/f, 1, 'last'));%The function of clipped waveform
subplot(2, 2, 2);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The waveform when harmonics is 2');
hold on
```

(b) The waveform for second harmonic

```
wave3 = amplitude *sin(2*pi*3*f*t); %The equation of Vt when harmonics is 3
sound(wave3, fs); %Making a sound
pause(1*duration);%Pause

%Plot the waveform when harmonics is 3
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave3(1:find(t <= T_max/f, 1, 'last'));%The function of clipped waveform
subplot(2, 2, 3);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The waveform when harmonics is 3');
hold on
```

(c) The waveform for third harmonic

```
wave4 = amplitude *(sin(2*pi*f*t)+sin(2*pi*2*f*t)+sin(2*pi*3*f*t)); %The equation of Vt as a chord
sound(wave4, fs); %Making a sound
pause(1*duration);%Pause

%Plot the chord waveform
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave4(1:find(t <= T_max/f, 1, 'last'));%The function of clipped waveform
subplot(2, 2, 4);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The chord waveform');
hold on
```

(d) The waveform for chord

Figure 47: The waveforms for different harmonics and their chord

According to the lab script, the T_{max} need to be set and the waveform need to clipped for convenience.

(c) **The final waveforms without phase shift**

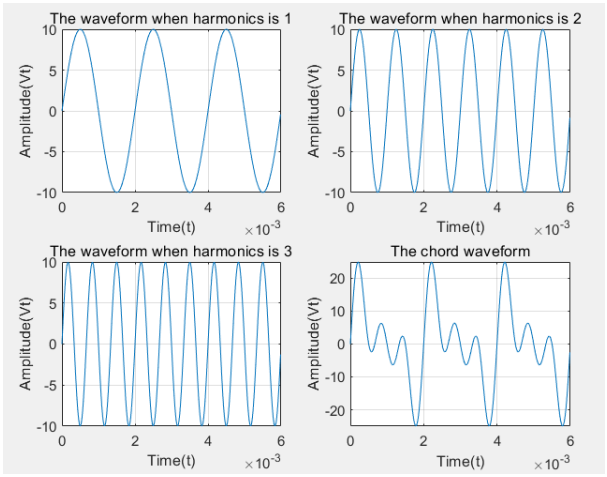


Figure 48: The final waveforms without phase shift

Since changing the harmonic means changing the frequency, as the harmonic increases, the period of each waveform will decrease. The waveform for chord is the sum of each waveform with different harmonic.

2. **Question E.2 Alter the phase of the third harmonic in equation 15 by 90 and repeat the tasks in the point above. [3 marks]**

After altering the phase, the third harmonic and the new chord will be:

(a) **The codes for third harmonic and chord**

```
wave3 = amplitude *sin(2*pi*3*f*t-pi/2); %The equation of Vt when harmonics is 3
sound(wave3, fs); %Making a sound
pause(1*duration);%Pause

%Plot the waveform when harmonics is 3
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave3(1:find(t <= T_max/f, 1, 'last'));%The function of clipped waveform
subplot(2, 2, 3);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The waveform when harmonics is 3');
hold on
```

(a) The codes for third harmonic with phase shift

```
wave4 = amplitude *(sin(2*pi*f*t)+sin(2*pi*2*f*t)+sin(2*pi*3*f*t-pi/2)); %The equation of Vt as a chord
sound(wave4, fs); %Making a sound
pause(1*duration);%Pause

%Plot the chord waveform
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave4(1:find(t <= T_max/f, 1, 'last'));%The function of clipped waveform
subplot(2, 2, 4);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The chord waveform');
hold on
```

(b) The codes for their chord

Figure 49: The codes for third harmonics and the chord

In the chord part, only the third harmonic has changed the phase and the final result is as follows:

(b) **The final waveforms with phase shift**

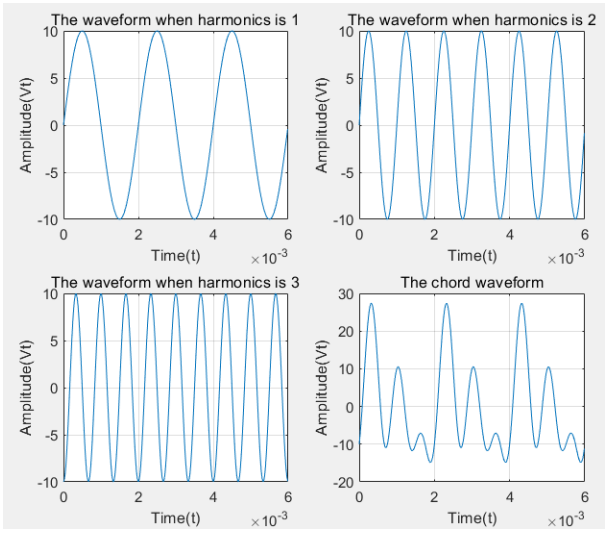


Figure 50: The final waveforms with phase shift

It can be seen from the figure that only the third harmonic changes the phase. Other harmonics remain unchanged. In addition, the waveform of chord has changed but not **significantly** and the sound produced is not very different.

3. **Question E.3**

Discuss the effect of altering the phase of the third harmonic, both on the sound and plot of the chord. Do the sound or plot change as you alter the

phase? Why? [2 marks]

(a) **The effect on the sound**

For the waveform with phase shift added, although it appears a little difference comparing with two sounds, the difference is not**obvious**. This is because the sustained sounds are not too loud and in this case, there are **no non-linear effects**. In addition, because this period of signal is repetitive and the duration is relatively short. Adding phase shift may change slightly. But according to the frequency of the repetition and the nature of the impulses, this change may be extremely weak to the listener [6].

For chord, it is also difficult to hear the difference with human ears. Although the total waveform looks different, as long as the volume is lowered, the non-linear effects are not important. Not only are the sounds that people hear very similar, but some people may think they are the same [6].

(b) **The effect on the plot of the chord**

By obtaining two sets of figures, it can be found that the two chords are different in shape, but the difference is not very obvious. The amplitude of each ripple is not the same. In addition, the maximum and minimum values of the waveform are not the same. When the phase shift is not set, the maximum and minimum values of the curve are 24.95V and -24.69V, respectively. When setting the phase shift, the maximum and minimum values of the curve are: 26.95V and -14.80V.

4. **Question E.4 What does the above tell you about the human ear? [2 marks]**

When the phase shift occurs, the shape of the waveform of chord will change. This phenomenon can be viewed in the result waveform above. In addition, the human ear can hear the sound of 20-20000 Hz, but the change of the sound (Because of phase shift) can not be sounded obviously. This is because if the volume is relatively low, the **non-linear effect** is not obvious.

And it is difficult for human ears to distinguish the sound after the phase shift. The final chord is produced by the sum of three different sets of sounds. And only the phase of the third harmonic is changed which means that the sounds of the other two harmonics have not changed. This will make the difference in sound even less obvious and the sound may be muted. This experiment tells me that it is difficult for human ears to clearly distinguish the difference in sound (Caused by phase shift).

References

[1] "Fourier analysis," https://www.investopedia.com/terms/f/fourieranalysis.asp, 2020.

[2] M. L´opez-Ben´ıtez, "Fourier synthesis of periodic waveforms," https://liverpool.instructure.com/courses/46000/pages/experiment-3?module_item_id=1258988, Department of Electrical Engineering Electronics, 2020.

[3] MathWorks, "Math, graphics, programming," https://ww2.mathworks.cn/products/matlab.html, 2021.

[4] S. W. Smith, "Fourier transform pairs," https://dspguide.com/ch11/4.htm, 2011.

[5] Electrical4U, "Band pass filter: Circuit transfer function (active passive)," https://www.electrical4u.com/band-pass-filter/, 2021.

[6] Physics of Music Notes, "Phase shifts and sounds," https://pages.mtu.edu/~suits/phaseshifts.html, 2021.

Appendices

A The overall codes for each part

A.1 Part A

Listing 1: Part A

```
%Assume f0 = 1000Hz and V0=1V
f0 = 1000;
V0=1;

t = 0:0.00001:0.002; %Define the time variable
harmonics =10; %In part A, the harmonic will change
ft_final = 0; %Initialize the function

%The process of generating the waveform
for n = 1:harmonics
 if mod(n,2) %Only consider the odd number
        ft =V0*sin(n*2*pi*f0*t)*(1/n); %The equation of ft
        ft_final = ft_final + ft; %Add them together
 end
end

%Plot the final waveform
figure(1);
subplot(2,1,1)
plot(t,ft_final);
grid on
xlabel('Time(t)') %Time
ylabel('Amplitude(ft)')
title('Resulting waveform')

%Plot the spectrum(frequency domain view)
for n = 1:harmonics
  if mod(n,2) %Only consider the odd number
     bn=1/n; %Because an=0,cn will be equal to bn
     X=n;
  end

%Plot discrete sequence data
subplot(2,1,2)
stem(X,bn)
title('Spectrum(frequency domain view)')
xlabel('f(w)') %Frequency
ylabel('Magnitude')
hold on
end
```

A.2 Part B

Listing 2: Part B

```
%Define the period and V
T = 1;
V = 1;
w0 = 2*pi; % w=2*pi/T

%Define an and bn
n_max = 50; %Define the max n
n = 1:n_max;
an = 0; a0 = 0;
bn = 2*V./(n*pi);

%Define the time variable
t = 0:1/50:3; %From 0s to 3s

%Initialize waveform f(t)
ft = zeros(size(t));

%The first 10 harmonics of this waveform
n = 1:10;
for i = 1:length(t)

ft(i) = sum(bn(n).*sin(n*w0*t(i)));

end

%Plot the first 10 harmonics of this waveform
figure(1)
subplot(2,2,1);
plot(t,ft);
title(['The first 10 harmonics of this waveform']);
xlabel('Time(t)');
ylabel('Amplitude(ft)');
grid on;


%Plot the original signal
x = sawtooth(2*pi*t,0);   %The original waveform
subplot(2,2,2);
plot(t,x);
xlabel('Time(t)');
ylabel('Amplitude(ft)');
grid on;
title('Original signal');

%The first 5 harmonics of this waveform
n = 1:5;
for i = 1:length(t)

ft(i) = sum(bn(n).*sin(n*w0*t(i)));

end

%Plot the first 5 harmonics of this waveform
subplot(2,2,3);
plot(t,ft);
title(['The first 5 harmonics of this waveform']);
xlabel('Time(t)');
```

```
ylabel('Amplitude(ft)');
grid on;
```

A.3 Part C

A.3.1 Part C (a)

Listing 3: Part C (a)

```
%Assume w0 = 2*pi, then T=1
w0 = 2*pi;

t = 0:0.001:3; %Time variable
harmonics =100; %In this part,the harmonics is not fixed
ft_final = 0; %Initialize the function

%The process of generating the waveform
for n = 1:harmonics
    if mod(n,2)%Only consider the odd number
        ft = (-1)^((n-1)/2)*sin(n*w0*t)/n^2; %The equation of ft
        ft_final = ft_final + ft; %Add them together
    end
end

%plot the resulting waveform
figure;
plot(t,ft_final);
grid on
xlabel('Time(t)')
ylabel('Amplitude(ft)')
title('Resulting waveform')
```

A.3.2 Part C (b)

Listing 4: Part C (b)

```
%Assume w0 = 2*pi, then T=1
w0 = 2*pi;

t = 0:0.0001:3; %Time variable
harmonics = 10; %In this part,the harmonics is not fixed
ft_final = 0; %Initialize the function

%The process of generating the waveform
for n = 1:harmonics
    if mod(n,2)%Only consider the odd number
        f1t= 0.1*cos((n+1)/2*w0*t);%The equation of ft
        ft_final= ft_final + f1t;%Add them together
    end
end

%Plot the resulting waveform
figure;
plot(t,ft_final);
grid on
xlabel('Time(t)')
ylabel('Amplitude(ft)')
title('Resulting waveform')
```

A.3.3 Part C (c)

Listing 5: Part C (c)

```
%Assume w0 = 2*pi, then T=1
w0 =2*pi;
t = 0:0.001:3; %Time variable

%The equation of ft
ft = sin(w0*t)-(4/(3*pi))*cos(2*w0*t)-(4/(15*pi))*cos(4*w0*t)-...
(4/(35*pi))*cos(6*w0*t)-(4/(63*pi))*cos(8*w0*t)-(4/(99*pi))*cos(10*w0*t);

%Plot the resulting waveform
figure;
plot(t,ft);
grid on
xlabel('Time(t)')
ylabel('Amplitude(ft)')
title('Resulting waveform')
```

A.3.4 Part C (d)

Listing 6: Part C (d)

```
%Assume w0 = 2*pi, then T=1
w0 = 2*pi;
t = 0:0.001:3; %Time variable

%The equation of ft
ft = -(4/(3*pi))*cos(2*w0.*t)-(4/(15*pi))*cos(4*w0.*t)-...
(4/(35*pi))*cos(6*w0.*t)-(4/(63*pi))*cos(8*w0.*t)-(4/(99*pi))*cos(10*w0.*t);


%Plot the resulting waveform
figure;
plot(t,ft);
grid on
xlabel('Time(t)')
ylabel('Amplitude(ft)')
title('Resulting waveform')
```

A.3.5 Part D (Original waveform)

Listing 7: Part D (Original waveform)

```
%assume f0 = 1000Hz
f0 = 1000;
t = 0:0.0000001:0.005; %Time variable
harmonics =10000; %In this part,the harmonics is not fixed
ft_final = 0; %Initialize the function

for n = 1:harmonics
     if mod(n,2)%Only consider the odd number
        ft =(3/pi)*sin(n*2*pi*f0*t)*(1/n);%The equation of ft
        ft_final = ft_final + ft;%Add them together
     end
end
os=overshoot(ft_final);%The percentage overshoot of the synthesised waveform

%plot the resulting waveform
figure;
plot(t,ft_final);
grid on
xlabel('Time(t)')
ylabel('Amplitude(ft)')
title('Resulting waveform')
```

A.3.6 Part D (a)

Listing 8: Function 5

```
%Assume f0 = 1000Hz
f0 = 1000;
t = 0:0.00001:0.005; %Time variable
harmonics = 3 %In this part,the harmonics is not fixed
ft_final = 0; %Initialize the function

for n = 1:harmonics
    if(harmonics<=10) %Low-pass filter
        if mod(n,2) %Only consider the odd number
         ft =(3/pi)*sin(n*2*pi*f0*t-(n*pi/2))*(1/n);%The equation of ft
         ft_final = ft_final + ft;%Add them together

         %plot the resulting waveform
         figure(1);
         plot(t,ft_final);
         grid on
         xlabel('Time(t)')
         ylabel('Amplitude(ft)')
         title('Resulting waveform')
        end

    else
       %The first 10 harmonics of this waveform
       ft_final = (3/pi)*(sin(2*pi*f0*t-pi/2)+1/3*sin(2*pi*3*f0*t-(3*pi/2))+...
           1/5*sin(2*pi*5*f0*t-(5*pi/2))+1/7*sin(2*pi*7*f0*t-(7*pi/2))+...
           1/9*sin(2*pi*9*f0*t-(9*pi/2)));

       %plot the resulting waveform
         figure(1);
         plot(t,ft_final);
         grid on
         xlabel('Time(t)')
         ylabel('Amplitude(ft)')
         title('Resulting waveform')
  end
end
```

A.3.7 Part D (b)

Listing 9: Part D (b)

```
%Assume f0 = 1000Hz
f0 = 1000;
t = 0:0.00001:0.005; %Time variable
harmonics = 3; %In this part,the harmonics is not fixed
ft_final = 0; %Initialize the function

for n = 1:harmonics
    if(harmonics<=10) %Low-pass filter
        if mod(n,2) %Only consider the odd number
         ft =(3/pi)*sin(n*2*pi*f0*t-(pi/2))*(1/n);%The equation of ft
         ft_final = ft_final + ft;%Add them together

         %plot the resulting waveform
         figure(1);
         plot(t,ft_final);
         grid on
         xlabel('Time(t)')
         ylabel('Amplitude(ft)')
         title('Resulting waveform')
        end

    else
       %The first 10 harmonics of this waveform
       ft_final = (3/pi)*(sin(2*pi*f0*t-pi/2)+1/3*sin(2*pi*3*f0*t-(pi/2))+...
       1/5*sin(2*pi*5*f0*t-(pi/2))+1/7*sin(2*pi*7*f0*t-(pi/2))+...
       1/9*sin(2*pi*9*f0*t-(pi/2)));

       %plot the resulting waveform
         figure(1);
         plot(t,ft_final);
         grid on
         xlabel('Time(t)')
         ylabel('Amplitude(ft)')
         title('Resulting waveform')
  end
end
```

A.3.8 Part D (c)

Listing 10: Part D (c)

```
%Assume f0 = 1000Hz
f0 = 1000;
t = 0:0.00001:0.005; %Time variable
harmonics = 5; %In this part,the harmonics is not fixed
ft_final = 0; %Initialize the function

for n = 1:harmonics
    if(harmonics<=10)%Low-pass filter
      switch(harmonics)
        case{1,2} %When the harmonics is 1 or 2
          ft_final=0;   %The equation of ft
        case{3,4} %When the harmonics is 3 or 4
          ft_final=(3/pi)*0.5*1/3*sin(2*pi*3*f0*t-((pi/2)*3)); %The equation of ft
        case{5,6} %When the harmonics is 5 or 6
          ft_final=(3/pi)*(0.5*1/3*sin(2*pi*3*f0*t-((pi/2)*3))+...%The equation of ft
          1/5*sin(2*pi*5*f0*t-((pi/2)*5)));
        case{7,8} %When the harmonics is 7 or 8
          ft_final=(3/pi)*(0.5*1/3*sin(2*pi*3*f0*t-((pi/2)*3))+...%The equation of ft
          1/5*sin(2*pi*5*f0*t-((pi/2)*5))+1/7*sin(2*pi*7*f0*t-((pi/2)*7)));
        case{9,10} %When the harmonics is 9 or 10
          ft_final=(3/pi)*(0.5*1/3*sin(2*pi*3*f0*t-((pi/2)*3))+...%The equation of ft
          1/5*sin(2*pi*5*f0*t-((pi/2)*5))+1/7*sin(2*pi*7*f0*t-((pi/2)*7))+...
          0.5*1/9*sin(2*pi*9*f0*t-((pi/2)*9)));
      end
    else
          ft_final=(3/pi)*(0.5*1/3*sin(2*pi*3*f0*t-((pi/2)*3))+...%The equation of ft
          1/5*sin(2*pi*5*f0*t-((pi/2)*5))+1/7*sin(2*pi*7*f0*t-((pi/2)*7))+...
          0.5*1/9*sin(2*pi*9*f0*t-((pi/2)*9)));
  end
end

%plot the resulting waveform
figure;
plot(t,ft_final);
grid on
xlabel('Time(t)')
ylabel('Amplitude(ft)')
title('Resulting waveform')
```

A.3.9 Part E

Listing 11: Part E

```
%When harmonics is 1
fs = 44100; %Sampling frequency
duration = 3; %Duration for 3 seconds
t = 0 : 1/fs : duration; %Time variable

%Assume amplitude = 10 and f = 500
amplitude = 10;
f = 500;

wave1 = amplitude *sin(2*pi*f*t); %The equation of Vt when harmonics is 1
sound(wave1, fs); %Making a sound
pause(1*duration);%Pause

%Plot the waveform when harmonics is 1
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave1(1:find(t <= T_max/f, 1, 'last')); %The function of clipped waveform
f1=figure
subplot(2,2,1);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The waveform when harmonics is 1');
hold on


%When harmonics is 2
fs = 44100; %Sampling frequency
duration = 3; %Duration for 3 seconds
t = 0 : 1/fs : duration; %Time variable

%Assume amplitude = 10 and f = 500
amplitude = 10;
f = 500;

wave2 = amplitude *(sin(2*pi*2*f*t)); %The equation of Vt when harmonics is 2
sound(wave2, fs); %Making a sound
pause(2*duration);%Pause

%Plot the waveform when harmonics is 2
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave2(1:find(t <= T_max/f, 1, 'last'));%The function of clipped waveform
subplot(2,2,2);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The waveform when harmonics is 2');
hold on

%When harmonics is 3
fs = 44100; %Sampling frequency
duration = 3; %Duration for 3 seconds
t = 0 : 1/fs : duration; %Time variable
```

```
%Assume amplitude = 10 and f = 500
amplitude = 10;
f = 500;

wave3 = amplitude *sin(2*pi*3*f*t); %The equation of Vt when harmonics is 3
sound(wave3, fs); %Making a sound
pause(1*duration);%Pause

%Plot the waveform when harmonics is 3
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave3(1:find(t <= T_max/f, 1, 'last'));%The function of clipped waveform
subplot(2,2,3);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The waveform when harmonics is 3');
hold on

%As a chord
fs = 44100; %Sampling frequency
duration = 3; %Duration for 3 seconds
t = 0 : 1/fs : duration; %Time variable

%Assume amplitude = 10 and f = 500
amplitude = 10;
f = 500;

wave4 = amplitude *(sin(2*pi*f*t)+sin(2*pi*2*f*t)+sin(2*pi*3*f*t)); %The equation of Vt as a ch
sound(wave4, fs); %Making a sound
pause(1*duration);%Pause

%Plot the chord waveform
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave4(1:find(t <= T_max/f, 1, 'last'));%The function of clipped waveform
subplot(2,2,4);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The chord waveform');
hold on



%Alter the phase of the third harmonic in equation 15 by 90 degree

%When harmonics is 1
fs = 44100; %Sampling frequency
duration = 3; %Duration for 3 seconds
t = 0 : 1/fs : duration; %Time variable

%Assume amplitude = 10 and f = 500
amplitude = 10;
f = 500;

wave1 = amplitude *sin(2*pi*f*t); %The equation of Vt when harmonics is 1
sound(wave1, fs); %Making a sound
```

```
pause(1*duration);%Pause

%Plot the waveform when harmonics is 1
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave1(1:find(t <= T_max/f, 1, 'last')); %The function of clipped waveform
f2=figure
subplot(2,2,1);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The waveform when harmonics is 1');
hold on


%When harmonics is 2
fs = 44100; %Sampling frequency
duration = 3; %Duration for 3 seconds
t = 0 : 1/fs : duration; %Time variable

%Assume amplitude = 10 and f = 500
amplitude = 10;
f = 500;

wave2 = amplitude *sin(2*pi*2*f*t); %The equation of Vt when harmonics is 2
sound(wave2, fs); %Making a sound
pause(2*duration);%Pause

%Plot the waveform when harmonics is 2
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave2(1:find(t <= T_max/f, 1, 'last'));%The function of clipped waveform
subplot(2,2,2);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The waveform when harmonics is 2');
hold on

%When harmonics is 3
fs = 44100; %Sampling frequency
duration = 3; %Duration for 3 seconds
t = 0 : 1/fs : duration; %Time variable

%Assume amplitude = 10 and f = 500
amplitude = 10;
f = 500;

wave3 = amplitude *sin(2*pi*3*f*t-pi/2); %The equation of Vt when harmonics is 3
sound(wave3, fs); %Making a sound
pause(1*duration);%Pause

%Plot the waveform when harmonics is 3
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave3(1:find(t <= T_max/f, 1, 'last'));%The function of clipped waveform
subplot(2,2,3);
plot(clipped_t, clipped_wave);
```

```
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The waveform when harmonics is 3');
hold on

%As a chord
fs = 44100; %Sampling frequency
duration = 3; %Duration for 3 seconds
t = 0 : 1/fs : duration; %Time variable

%Assume amplitude = 10 and f = 500
amplitude = 10;
f = 500;

wave4 = amplitude *(sin(2*pi*f*t)+sin(2*pi*2*f*t)+sin(2*pi*3*f*t-pi/2)); %The equation of Vt as
sound(wave4, fs); %Making a sound
pause(1*duration);%Pause

%Plot the chord waveform
T_max = 3;
clipped_t = t(1:find(t <= T_max/f, 1, 'last')); %The time of clipped waveform
clipped_wave = wave4(1:find(t <= T_max/f, 1, 'last'));%The function of clipped waveform
subplot(2,2,4);
plot(clipped_t, clipped_wave);
grid on
xlabel('Time(t)');
ylabel('Amplitude(Vt)');
title('The chord waveform');
hold on
```