ELEC 2430 Communications Theory MATLAB Laboratory Session

First A. Huo Yuheng, *Student*, *SWJTU*

All the files needed can be find
https://github.com/Neowless/Solution-to-the-XJEL-2430-Communications-Theory-MATLAB-Homeworks
Or scan the QR code

I. Fourier Series

This task aims to compare the difference of different terms of Fourier expansion, which are added together to generate a square wave signal.

A. Answer

With the increment of the value of N, which represents the terms of the whole function, the precision of the fitting degree goes higher.

The original square wave function can be expressed as:

B. Script

```
close all
clear all
x = -2.5:0.01:2.5;
% set the range of x between -2.5 and 2.5
y = linspace(0.5,0.5,501);
% set the gap between each x equals 0.01,
and initialize the y = 0.5 for
% a0 = 0.5
o = round((cos(x*pi)+1)/2);
% biuld the original square wave function
N = input('Please input the number of
N.')
for i = 1:N
    % assuming that it iterates  120
times and save the image of each
    % iteration function
    a = 2*sin(i*pi/2)/i/pi;
    y = y + a*cos(i*pi*x);
    % b =  0 can be proved by calculation
end
figure(1)
plot(x,y,'-r')
hold
plot(x,o,'-b')
hold
% set the colour of each lines
set(gca,'XLim',[-2.5 2.5]);
set(gca,'YLim',[0 2]);
% set the range of the image
```

C. Figures

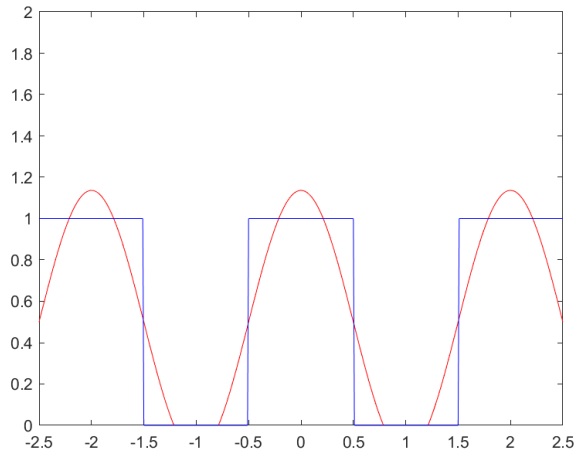


Fig.1. N equals 1, the whole function appears as a simple sin wave.

Huo Yuheng., is with the Leeds Joint School South-West Jiaotong University, Chengdu, Sichuan 61000 China. (e-mail: el17yh@leeds.ac.uk).

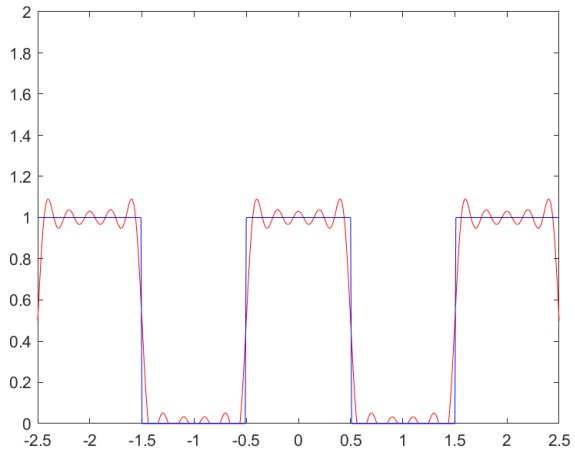


Fig.2. N equals 10, the whole function losses a lot of details.

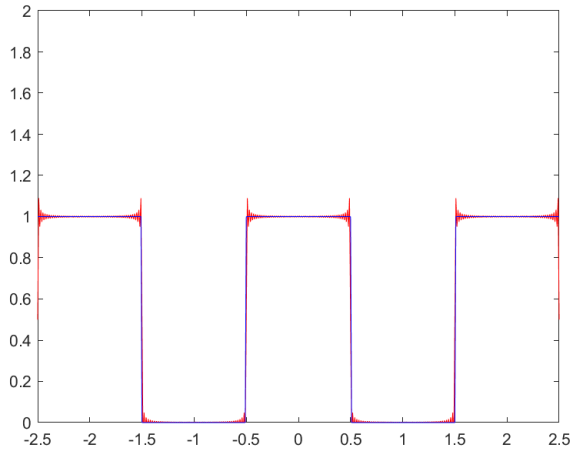


Fig.3. N equals 100, the whole function mostly reflect the shape of the original square wave, but there always many ‘peaks’ at the edges.

.

II. USING MATLAB TO CALCULATE THE FFT OF A SINE WAVE

A. Increasing the number of samples, with a constant sampling rate

1) Answer

With the increment of the samples with a constant sampling rate, and the number of FFT points (N) is set as 4096 which determines the resolution of the frequency domain to easily reflect the phenomenon.

The original signal frequency is 50Hz, and the sampling frequency is 500Hz, which is far from Nyquist frequency.

In general, the value at 50Hz in the frequency domain, is getting larger with the increment of the samples, the amplitude spectrum in the frequency domain appears as zooms out horizontally, and the amplitude of these spectrum almost remains at constant values.

2) Script

```
% This program aims to process a sin
windows function
% N is the number of the sampling points
N = input('Please input the number of
sampling points.')
Fs = 500;
F = 50;
% F is the frequency of the original
signal
Tg = 1/ Fs;
% sample frequency @ 500hz, Tg means the
time gap between each sample point
resolution = 4096;
% set te resolution in the time domain
t = linspace(0,N*Tg,N+1);
ti = linspace(0,N*Tg,N*Tg*1000);
% calculate the time at each point, t is
time of sampling, ti is built to
% show the orginal signal
L = length(t);
S = 0.7*sin(2*pi*50*t);
si = 0.7*sin(2*pi*50*ti);
% construct the original function
figure(1);
subplot(2,1,1)
plot(t,S)
hold
plot(ti,si)
title('Original Signal')
xlabel('t (seconds)')
ylabel('X(t)')
set(gca,'XLim',[0 N/Fs]);
set(gca,'YLim',[-1.2 1.2]);
% print the original image of the singnal
Y = fft(S,resolution);
P2 = abs(Y);
i=1:resolution/2;
f=(i-1)*Fs/resolution;
yy=zeros(resolution/2,1);
yy(i)=P2(i);
yy=yy/(L/2);
yy(1)=yy(1)/2;
subplot(2,1,2)
plot(f,yy)
title('Single-Sided Amplitude Spectrum of
S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
set(gca,'XLim',[0 Fs/2]);
set(gca,'YLim',[0 1]);
```

3) Figures

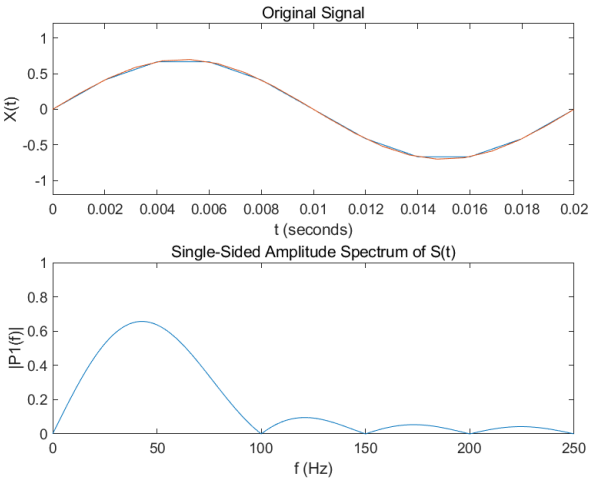


Fig.1. N equals 10, the whole function appears as a random wave.

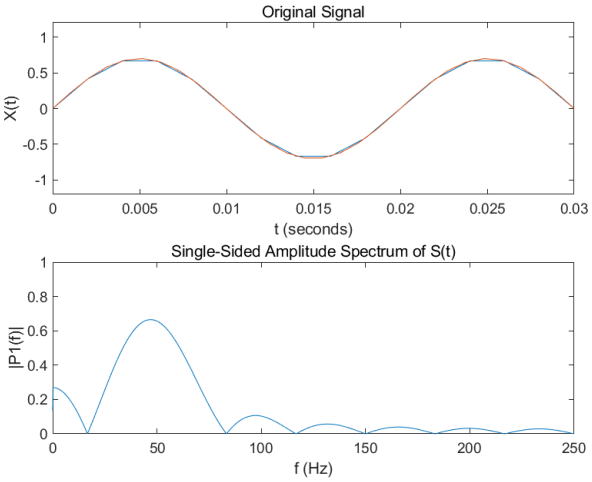


Fig.2. N equals 15, the whole function is zooms horizontally.

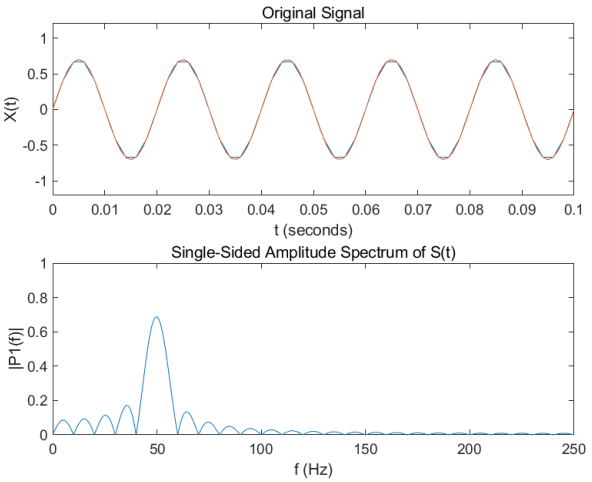


Fig.3. N equals 50, the whole function starts appears as a sinc function.

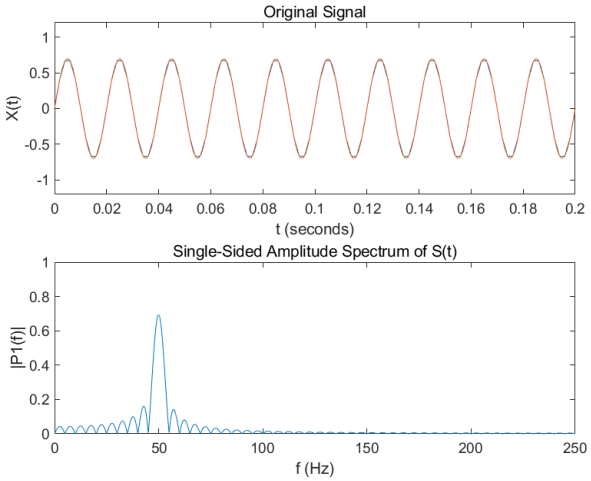


Fig.1. N equals 100, the whole function appears as a sinc function.

B. The number of FFT points (N) determines the resolution of the frequency domain plot Other Recommendations

1) Answer

All the answer is under the condition that the sampling frequency is remarkably large.

The original signal frequency is 50Hz, and the sampling frequency is 500Hz, and the number of the sampling points is 5000, which means there are 500 periods in the windows function.

If N is smaller than the number of the sampling points n, the spectrum in the frequency domain shows like a isosceles triangle, with the increment of N, the height of it increases.

If N is exact same as n, the quality of the spectrum in the frequency domain appears as the best quality.

If N is greater than n, the spectrum starts to becomes a sinc function.

2) Script

```
% This program aims to process a sin
windows function
% N is the number of the sampling points
for r = 1:1:200
N = 5000;
Fs = 500;
F = 50;
% F is the frequency of the original
signal
Tg = 1/ Fs;
% sample frequency @ 500hz, Tg means the
time gap between each sample point
resolution = r*50;
% set te resolution in the time domain
t = linspace(0,N*Tg,N+1);
ti = linspace(0,N*Tg,N*Tg*1000);
% calculate the time at each point, t is
time of sampling, ti is built to
% show the orginal signal
L = length(t);
S = 0.7*sin(2*pi*50*t);
si = 0.7*sin(2*pi*50*ti);
% construct the original function
figure(1);
subplot(2,1,1)
```

```
plot(t,S)
hold
plot(ti,si)
title('Original Signal')
xlabel('t (seconds)')
ylabel('X(t)')
set(gca,'XLim',[0 N/Fs]);
set(gca,'YLim',[-1.2 1.2]);
% print the original image of the singnal
Y = fft(S,resolution);
P2 = abs(Y);
i=1:resolution/2;
f=(i-1)*Fs/resolution;
yy=zeros(floor(resolution/2),1);
yy(i)=P2(i);
yy=yy/(L/2);
yy(1)=yy(1)/2;
subplot(2,1,2)
plot(f,yy)
title('Single-Sided Amplitude Spectrum of
S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
set(gca,'XLim',[0 Fs/2]);
set(gca,'YLim',[0 1]);
str1 = 'sinc_fft';
str2 = num2str(N);
str3 = '.png'
name = [str1,str2,str3]
end
```

C. In line with Nyquist's sampling theorem, to correctly reconstruct a particular signal, the sampling frequency must be at least twice the maximum frequency component of the signal.

1) Answer

For the original signal, the sampling frequency shall be greater than or equal to twice the highest frequency component of the sampled signal (preferably greater than twice, and some special cases may occur when it is equal to twice).If a continuous sinusoidal signal by frequency for 10 Hz, 40 Hz and 80 Hz three sinusoidal component of the signal, so if we want to sampling of the signal, and makes the sampling of the signal points to represent the original signal waveform, we need to ensure that the sampling frequency is greater than or equal . so can make sure with the sampling of the signal points can back to the original signal reconstruction. For the real signal to be recovered without distortion, the sampling frequency must be greater than twice the maximum frequency of the signal. When sampling a signal with the sampling frequency F, the frequency above F/2 in the signal does not disappear, but the symmetric image is superimposed with the original frequency components below F/2 in the frequency band below F/2. This phenomenon is called "aliasing".

However, this is also a liner dynamic phenomenon, with the increment of the sampling frequency, you can clearly find out that the spectrum is shifting in the frequency domain.

2) Script

```
% This program aims to process a sin
windows function
% N is the number of the sampling points
ff = input('Please input the number of
sampling frequency.')
N = ff*5;
Fs = ff;
F = 50;
% F is the frequency of the original
signal
Tg = 1/ Fs;
% sample frequency @ 500hz, Tg means the
time gap between each sample point
resolution = 4096;
% set te resolution in the time domain
t = linspace(0,N*Tg,N+1);
ti = linspace(0,N*Tg,N*Tg*1000);
% calculate the time at each point, t is
time of sampling, ti is built to
% show the orginal signal
L = length(t);
S =
0.2*sin(2*pi*10*t)+0.3*sin(2*pi*40*t)+0.4
*sin(2*pi*80*t);
si =
0.2*sin(2*pi*10*ti)+0.3*sin(2*pi*40*ti)+0
.4*sin(2*pi*80*ti);
% construct the original function
figure(1);
subplot(2,1,1)
plot(t,S)
hold
plot(ti,si)
title('Original Signal')
xlabel('t (seconds)')
ylabel('X(t)')
set(gca,'XLim',[0 N/Fs]);
set(gca,'YLim',[-1.2 1.2]);
% print the original image of the singnal
Y = fft(S,resolution);
P2 = abs(Y);
i=1:resolution/2;
f=(i-1)*Fs/resolution;
yy=zeros(floor(resolution/2),1);
yy(i)=P2(i);
yy=yy/(L/2);
yy(1)=yy(1)/2;
subplot(2,1,2)
plot(f,yy)
title('Single-Sided Amplitude Spectrum of
S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
set(gca,'XLim',[0 Fs/2]);
set(gca,'YLim',[0 1]);
```

3) Figures

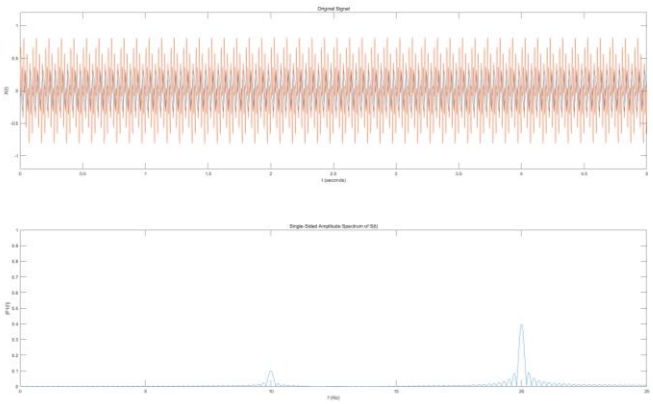


Fig.1. ff equals 50, you can find the spectrum of 10 Hz, but the spectrums for 40Hz and 80 Hz are totally missing

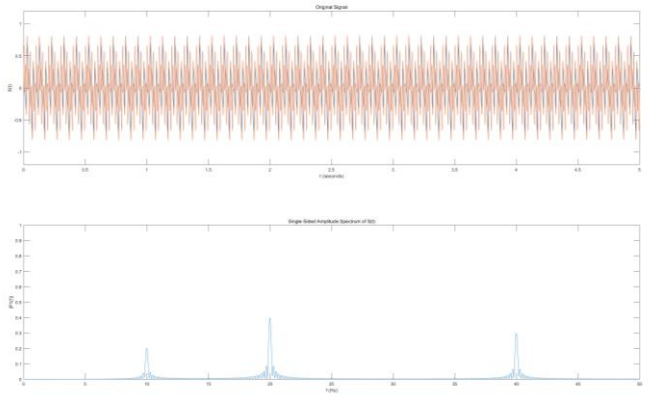


Fig.2. ff equals 100, you can find the spectrums of 10 Hz and 40Hz, but the spectrum for 80 Hz is totally wrong

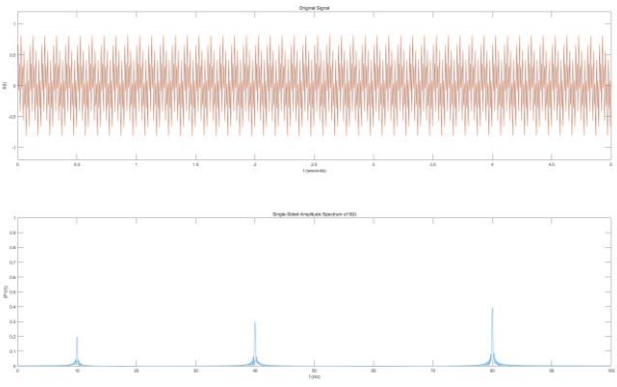


Fig.2. ff equals 200, you can clearly find the spectrums of 10 Hz and 40Hz and 80 Hz

D. If a signal is buried in noise, it can be difficult to identify the frequency components in the time-domain.

1) Answer

A noise-covered signal is difficult to observe in the time domain, so converting it to the frequency domain can help us observe the frequency of its primary constituent. In this case, our baseband data is made up of a periodic sin waves.

2) Script

```
% This program aims to process a sin
windows function
% N is the number of the sampling points
ff = 200;
N = ff*5;
Fs = ff;
F = 50;
% F is the frequency of the original
signal
Tg = 1/ Fs;
% sample frequency @ 500hz, Tg means the
time gap between each sample point
resolution = 4096;
% set te resolution in the time domain
t = linspace(0,N*Tg,N+1);
% calculate the time at each point, t is
time of sampling, ti is built to
% show the orginal signal
L = length(t);
S =
0.2*sin(2*pi*10*t)+0.3*sin(2*pi*40*t)+0.4
*sin(2*pi*80*t);
S = S + randn(size(t));
% construct the original function
figure(1);
subplot(2,1,1)
plot(t,S)
hold
title('Original Signal')
xlabel('t (seconds)')
ylabel('X(t)')
set(gca,'XLim',[0 N/Fs]);
set(gca,'YLim',[-3 3]);
% print the original image of the singnal
Y = fft(S,resolution);
P2 = abs(Y);
i=1:resolution/2;
f=(i-1)*Fs/resolution;
yy=zeros(floor(resolution/2),1);
yy(i)=P2(i);
yy=yy/(L/2);
yy(1)=yy(1)/2;
subplot(2,1,2)
plot(f,yy)
title('Single-Sided Amplitude Spectrum of
S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
set(gca,'XLim',[0 Fs/2]);
set(gca,'YLim',[0 1]);
```

3) Figures

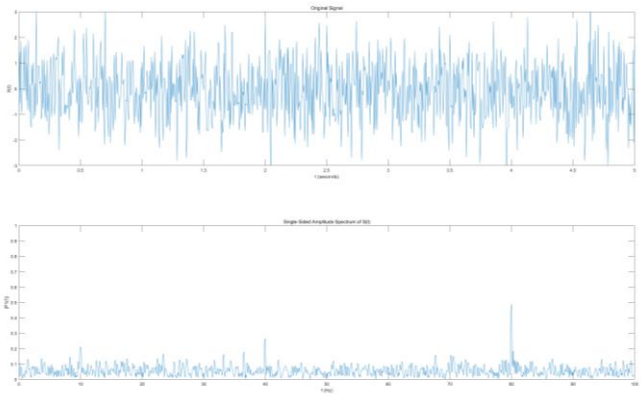


Fig.1. It is clear that the spectrum has got peaks @ 10Hz, 40Hz and 80Hz

III. THE FREQUENCY DOMAIN REPRESENTATION OF A SQUARE WAVE

A. In the frequency spectrum of a square wave, power is contained in the odd harmonics of the fundamental frequency.

1) Answer

From the problem we did in question2, the square function can use Fourier function to devote into N sin waveforms, and I also did the reverse process. With different values of N, different terms of Fourier series are added to generate a signal which should be the same with the square wave if N is sufficiently big. And you also see that with the N becomes bigger, the amplitude becomes smaller.

2) Script

```
% This program aims to process a sin
windows function
% N is the number of the sampling points
N = 5000;
Fs = 500;
F = 50;
% F is the frequency of the original
signal
Tg = 1/ Fs;
% sample frequency @ 500hz, Tg means the
time gap between each sample point
resolution = 4096;
% set te resolution in the time domain
t = linspace(0,N*Tg,N+1);
ti = linspace(0,N*Tg,N*Tg*1000);
% calculate the time at each point, t is
time of sampling, ti is built to
% show the orginal signal
L = length(t);
S = floor(sin(2*pi*F*t)/2+1)*2-1;
si = floor(sin(2*pi*F*ti)/2+1)*2-1;
% construct the original function
figure(1);
subplot(2,1,1)
plot(t,S)
hold
plot(ti,si)
title('Original Signal')
xlabel('t (seconds)')
ylabel('X(t)')
set(gca,'XLim',[0 N/Fs]);
set(gca,'YLim',[-1.2 1.2]);
% print the original image of the singnal
Y = fft(S,resolution);
P2 = abs(Y);
i=1:resolution/2;
f=(i-1)*Fs/resolution;
yy=zeros(floor(resolution/2),1);
yy(i)=P2(i);
yy=yy/(L/2);
yy(1)=yy(1)/2;
subplot(2,1,2)
plot(f,yy)
title('Single-Sided Amplitude Spectrum of
S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
set(gca,'XLim',[0 Fs/2]);
set(gca,'YLim',[0 1]);
```

3) Figures

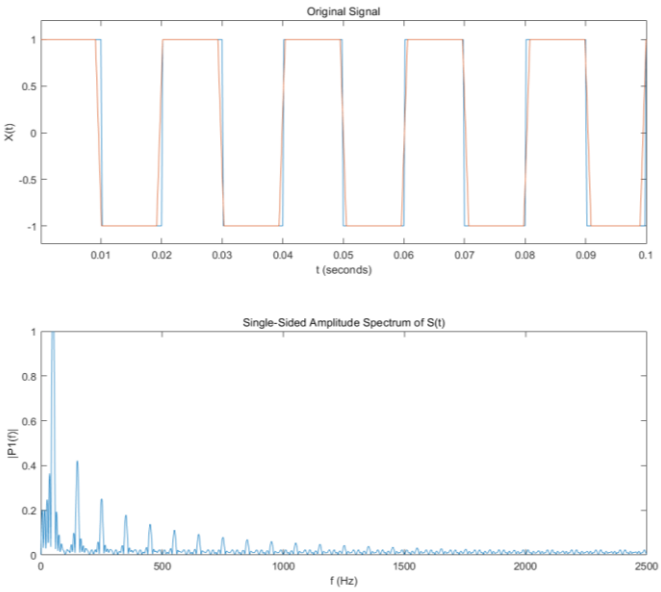


Fig.2. ff equals 200, you can clearly find the spectrums of 50Hz, and the sub-spectrum of it

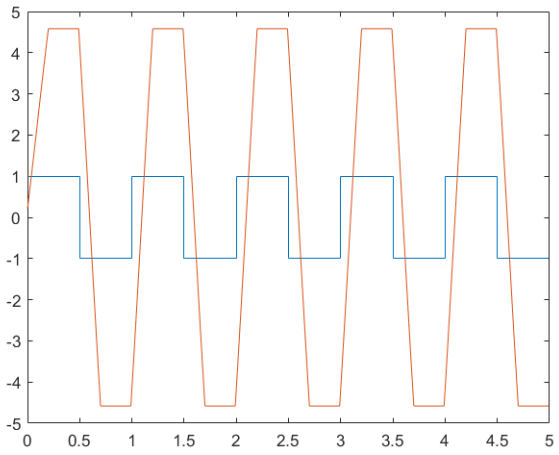
B. Although the power is concentrated mainly in the first four of five harmonics, a perfect square wave has an infinite frequency bandwidth. This is the reason why square waves are often 'rounded' off using a raised cosine filter, which removes the sharp corners that contribute the higher-order harmonics. This reduces the bandwidth and can help to minimise inter-symbol interference.

1) Script

```
fs = 100;
```

```
Tg = 1/ fs;
L = 500;
n = 0:L-1;
t = Tg*n;
y = square(2*pi*t);
rolloff = 0.3;
span = 0.4;
sps = 50;
b = rcosdesign(rolloff,span,sps,'sqrt');
x = upfirdn(y,b);
stairs(t,y)
hold on
plot(t,x(1:500))
```

2) Figures



C. Plot of random bit stream.

1) Answer

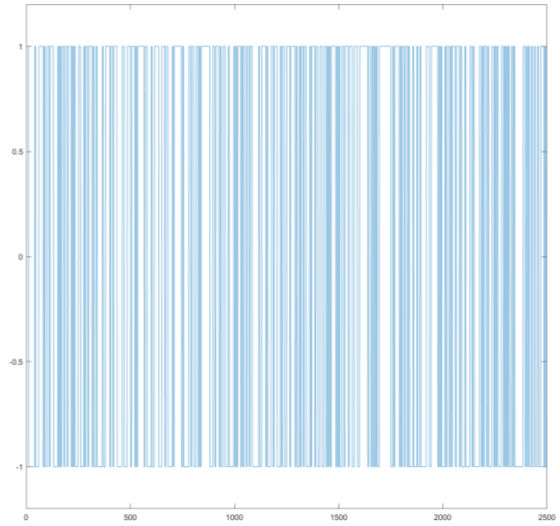


Fig.1.

```
>>randombitstream(500,5);
>> x = 1:1:2500;
>>plot(x,ans)
>> set(gca,'YLim',[-1.2 1.2]);
```

2) Script

```
% This function is to form a random bit
stream(±1)
% The first value should be the number of
original signal bits
% The second value should be the value of
the repeating times
% This is not a 100% random bit stream,
if you want it to be totally random
% , please delete rng(100);
function [n] = randombitstream(l,bw)
rng(100);
o = (round(rand(l)));
o = o(1,:);
o = 2.*o-1;
for i = 1:1:length(o)
    for x =1:1:bw
        n(i*bw-bw+x) = o(i);
    end
end
end
```

D. When the square wave is replaced by a random bit stream, the odd-harmonics of the square wave are replaced by a sinc-like frequency spectrum. The nulls in this sinc function appear at the multiples of the fundamental frequency.

1) Answer

We can see the difference between square waveform and the bitstream. the odd-harmonics of the square wave are replaced by a sinc-like frequency spectrum. In complex periodic oscillations, fundamental and harmonic waves are involved in. Sine wave component equal to the longest period of the oscillation is called the fundamental wave. The frequency corresponding to this period is called the fundamental frequency.

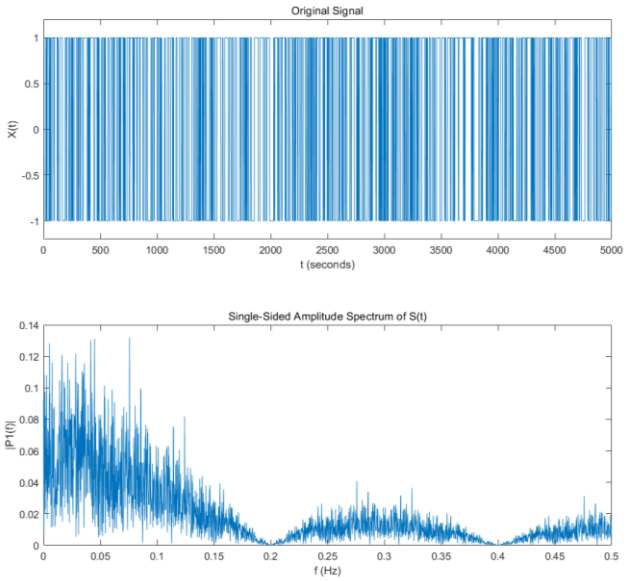
2) Script

```
n = 1000;
% The number of the sampling points in
the time domain
bt = 5;
% The original singnal generating period
bt
bw = 1/bt;
% The original singnal generating
frequency fs
stream = randombitstream(n,bt);
% The sampling frequency remains 1 Hz
t = 1:1:n*bt;
% calculate the time at each point
L = length(t);
S = stream;
% construct the original function
figure(1);
subplot(2,1,1)
plot(t,S)
hold
```

```
title('Original Signal')
xlabel('t (seconds)')
ylabel('X(t)')
set(gca,'XLim',[0 n*bt]);
set(gca,'YLim',[-1.2 1.2]);
% print the original image of the singnal
Y = fft(S);
P2 = abs(Y/L);
P1 = P2(1:floor(L/2+1));
P1(2:end-1) = 2*P1(2:end-1);
f = 1*(0:(L/2))/L;


Y = fft(S);
P2 = abs(Y/L);
P1 = P2(1:floor(L/2+1));
P1(2:end-1) = 2*P1(2:end-1);
subplot(2,1,2)
plot(f,P1)
title('Single-Sided Amplitude Spectrum of S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
set(gca,'XLim',[0 0.5]);
```

3) Figure



E. By comparing the spectrum of a square wave to that of a random bit stream, it can be seen power is distributed more evenly across the band in the random bit stream case, as opposed to at specific harmonics in the square wave case.

1) Answer

By comparing, it is clear to see the spectrums of a square wave to that of a random bit stream, it can be seen power is distributed more evenly across the band in the random bit stream case

2) Script

```
% This program aims to process a sin
windows function
% N is the number of the sampling points
N = 1000;
Fs = 1;
F = 0.2;
% F is the frequency of the original
signal
Tg = 1/ Fs;
% sample frequency @ 500hz, Tg means the
time gap between each sample point
resolution = 4096;
% set te resolution in the time domain
t = linspace(0,N*Tg,N+1);
ti = linspace(0,N*Tg,N*Tg*1000);
% calculate the time at each point, t is
time of sampling, ti is built to
% show the orginal signal
L = length(t);
S = floor(sin(2*pi*F*t)/2+1)*2-1;
si = floor(sin(2*pi*F*ti)/2+1)*2-1;
% construct the original function
figure(1);
subplot(2,2,1)
plot(t,S)
hold
plot(ti,si)
title('Original Signal')
xlabel('t (seconds)')
ylabel('X(t)')
set(gca,'XLim',[0 N/Fs]);
set(gca,'YLim',[-1.2 1.2]);
% print the original image of the singnal
Y = fft(S,resolution);
P2 = abs(Y);
i=1:resolution/2;
f=(i-1)*Fs/resolution;
yy=zeros(floor(resolution/2),1);
yy(i)=P2(i);
yy=yy/(L/2);
yy(1)=yy(1)/2;
subplot(2,2,2)
plot(f,yy)
title('Single-Sided Amplitude Spectrum of
S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
set(gca,'XLim',[0 Fs/2]);
set(gca,'YLim',[0 1]);

n = 1000;
% The number of the sampling points in
the time domain
bt = 5;
% The original singnal generating period
bt
bw = 1/bt;
% The original singnal generating
frequency fs
stream = randombitstream(n,bt);
```

```
% The sampling frequency remains 1 Hz
t = 1:1:n*bt;
% calculate the time at each point
L = length(t);
S = stream;
% construct the original function
hold
figure(1);
subplot(2,2,3)
plot(t,S)
hold
title('Original Signal')
xlabel('t (seconds)')
ylabel('X(t)')
set(gca,'XLim',[0 n*bt]);
set(gca,'YLim',[-1.2 2.2]);
% print the original image of the singnal
Y = fft(S);
P2 = abs(Y/L);
P1 = P2(1:floor(L/2+1));
P1(2:end-1) = 2*P1(2:end-1);
f = 1*(0:(L/2))/L;


Y = fft(S);
P2 = abs(Y/L);
P1 = P2(1:floor(L/2+1));
P1(2:end-1) = 2*P1(2:end-1);
subplot(2,2,4)
plot(f,P1)
title('Single-Sided Amplitude Spectrum of
S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
set(gca,'XLim',[0 0.5]);
```

3) Figure