Fast Fourier transform of a given Signal

1. Source Code

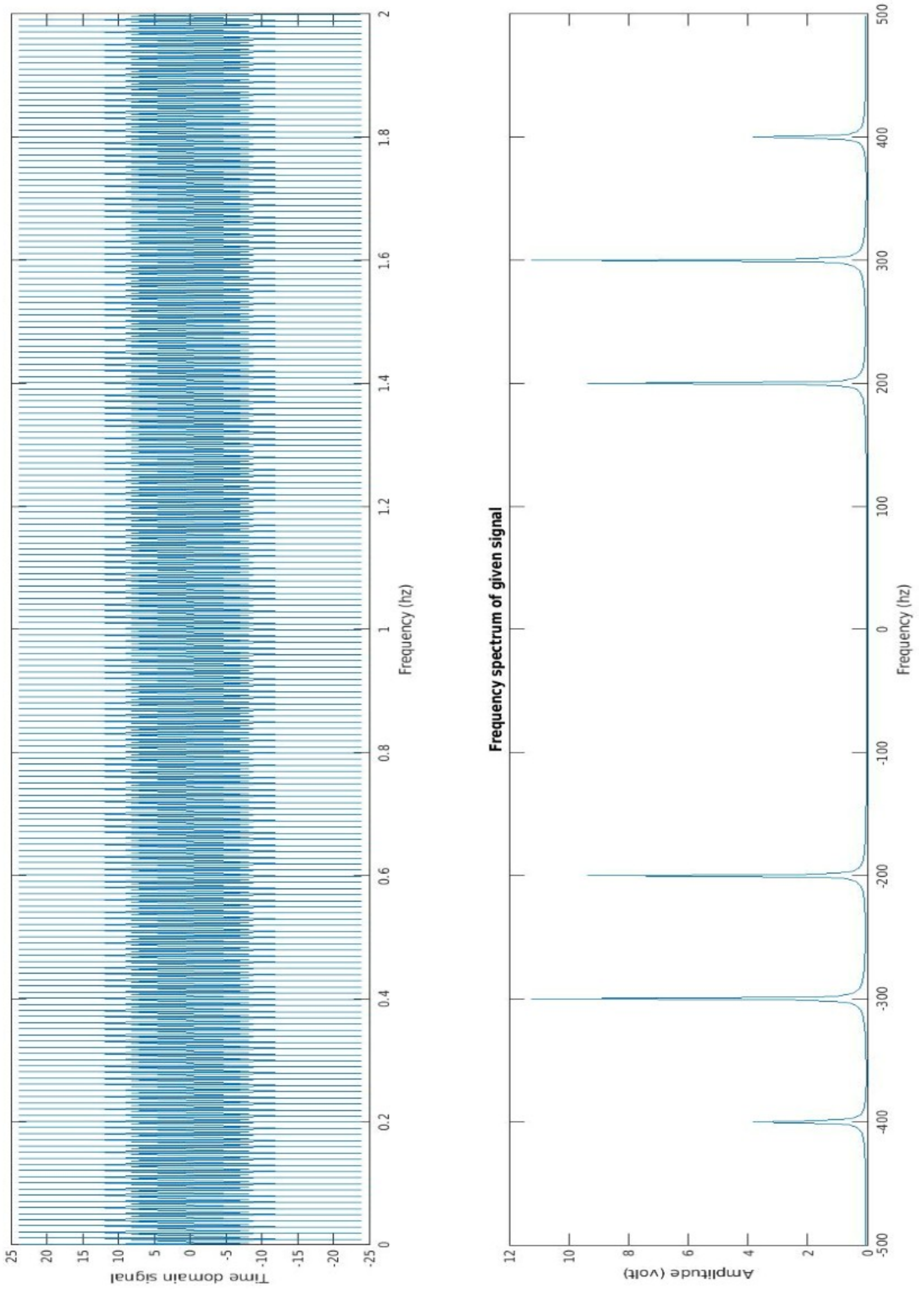
```
clc;
clear all;
fs = 1000;
ts = 1/fs;
t = 0:ts:2-ts;
y = 10 * sin(2*pi*200*t) + 5 * sin(2*pi*400*t) + 12 *
sin(2*pi*300*t);
N = 2^nextpow2(fs);

% fast fourier transform functions
yy  = fft(y,N);
yyy = fftshift (yy);
f = fs * (-N/2: N/2 -1)/N ;

% Plot graphs
subplot(2,1,1);
plot(t,y)
xlabel('Frequency (hz)');
ylabel('Time domain signal');

subplot(2,1,2);
plot(f, (2 * abs(yyy/N)));
title('Frequency spectrum of given signal');
xlabel('Frequency (hz)');
ylabel('Amplitude (volt)');
```

2. Observation



Design an IIR filter

1. Source Code

```
clc;
clear all;
fs = 100e3;
f  = 5e3;
ts = 1/fs;
t  = 0:ts: 5e-3-ts
X  = 5 * sin (2* pi* f * t);
Z  = awgn(X,1);

% plot the output - sinusoidal signal
plot(t,X);
title('Sinusoidal signal');

% Plot 2 - signal with noise
plot(1,Z);
title('Signal with noise');

nfft  = length(Z)
nfft2 = 2^nestpow2(nfft);

% fast fourier transformation
Fy = fft(Z, nfft2);
Fy = Fy(1:nfft2/2);

xfft = fs * (0:nfft2/2 - 1)/nfft2;

% plot3
plot(xfft, abs( Fy/max(Fy));
0:40;
wc = 2* pi * f/fs;
[b,a] = butter(0, wc, 'low');
x_f_iir = filter(b,a,Z);

figure;

plot(t,x_f_iir);
title('Filtered Sinusoidal Wave');
```

Implement low pass filter

1. Source Code

```
Fs = 100;
T  = 1/Fs;
t  = 0:T:1-T;
% generate signal
s  = sin(2 * pi * 10 * t);
% generate noise
noise = 0.5 * randn(size(t));
% input signal with random noise
x = s + noise;

plot(x)
shg
plot(x)

% low pass filter
d = designfilt('lowpassfir','FilterOrder',5,'CutOffFrequency',11,'SampleRate',Fs);

% output - filtered waveform
y = filter(d,x)

plot(y)

plot(X), hold on; plot(y)
```