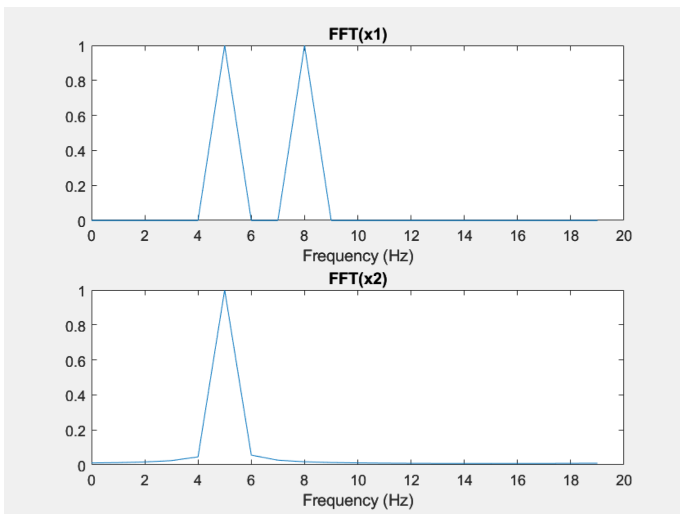
Narges Sadat Seyed Haeri – 810100165

Signals and Systems

CA5 Report

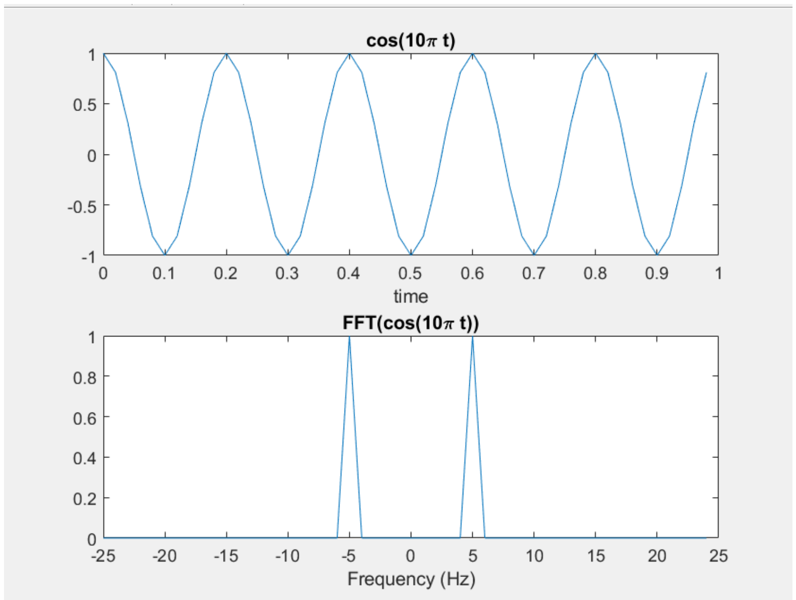
- ***Part 1:***

1.0) As stated in the instructions, when considering the frequencies of the exp function as 5 and 8, two distinct peaks are clearly visible at 5 and 8 Hz. However, if we consider these values as 5 and 5.1, since the difference between them is less than the frequency resolution (1 Hz), only one peak with some noise is observable. This is illustrated in the following image:



1.1) This code generates and analyzes a cosine signal with a frequency of 5 Hz. The sampling frequency is set to 50 Hz, and the time vector t is created with a time step of 1/50 seconds. The code then generates the cosine signal x1 = $\cos(10\pi t)$ and plots it in the first subplot. The second subplot displays the centered (using fftshift) magnitude spectrum of the signal obtained through the Fast Fourier Transform (FFT), showcasing the frequency content of the signal.

```
fs = 50;
t_start = 0;
t_end = 1;
ts = 1 / fs;
t = t_start:ts:t_end - ts;
N = length(t);
f = (-fs / 2):(fs / N):(fs / 2 - fs / N);
x1 = cos(2 * pi * 5 * t);
figure
subplot(2, 1, 1)
plot(t, x1)
title('cos(10\pi t)')
xlabel('time')
x1_fft = fftshift(fft(x1));
y1 = abs(x1_fft) / max(abs(x1_fft));
subplot(2, 1, 2)
plot(f,y1 )
xlabel('Frequency (Hz)')
title('FFT(cos(10\pi t))')
```



1.2) The Fourier transform of a cosine signal $cos(\omega 0\ t)$ is known to be expressed as:

$$\mathcal{F}\{\cos(\omega_0 t)\} = \pi\delta(\omega - \omega_0) + \pi\delta(\omega + \omega_0)$$

When normalized in MATLAB, the (π) coefficients are removed from the result. Substituting the specific value $(\omega_0 = 10\pi)$, the normalized Fourier transform becomes

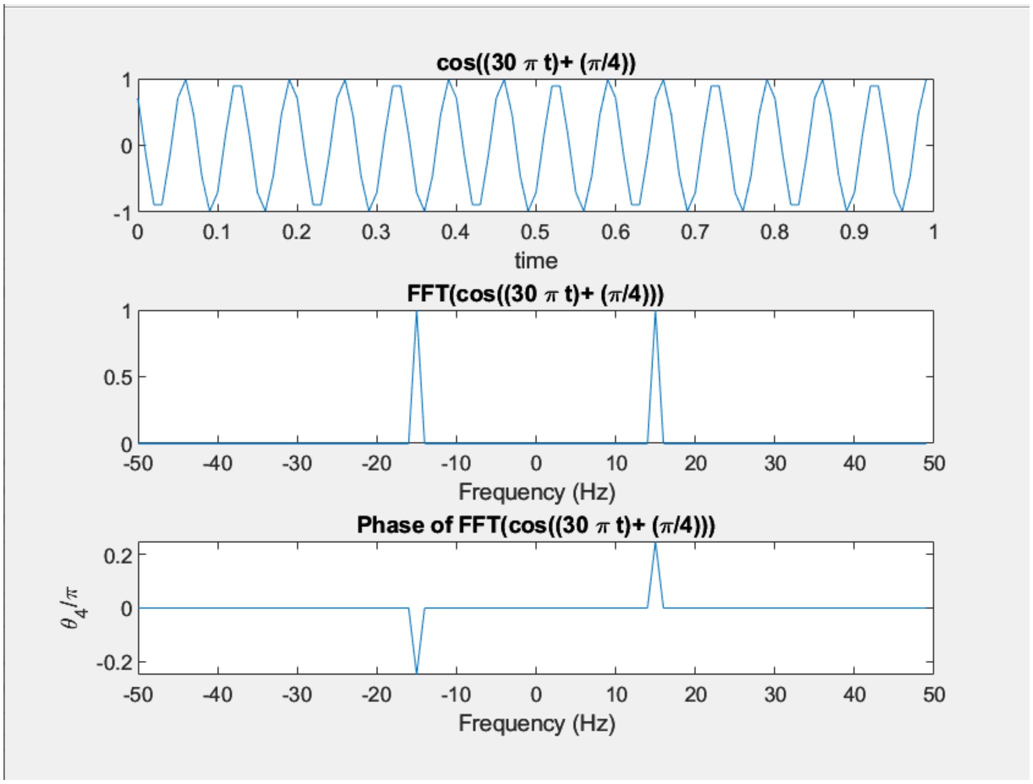
$$\mathcal{F}_N\{\cos(10\pi t)\} = \delta(\omega - 10\pi) + \delta(\omega + 10\pi)$$

Additionally, considering that the plots are in terms of frequency f rather than (ω), the variable transformation $(\omega = 2\pi f)$ is applied, resulting in

$$\mathcal{F}_N\{\cos(10\pi t)\} = \delta(f - 5) + \delta(f + 5)$$

As observed, the theoretical calculations align with the obtained values.

2) In this code we generate, analyze, and visualize the properties of a cosine signal with a frequency of 30π Hz and a phase shift of π/4. The signal is sampled with a sampling frequency of 100 Hz over a time period from 0 to 1 second. The first subplot displays the original time-domain signal, and the second subplot illustrates the magnitude spectrum obtained through the Fast Fourier Transform (FFT), showing the frequency components of the signal. In the third subplot, the phase information of the FFT is extracted, and small values are thresholded to zero to improve clarity. The phase is then normalized by π and plotted against frequency. Overall, the code provides insights into both the time and frequency characteristics of the given cosine signal, including its phase information in the frequency domain.



To obtain the Fourier transform of the signal ($\cos\left(30\pi t + \frac{\pi}{4}\right)$),we start with the expression:

$$[\mathcal{F}\{\cos\left(30\pi t + \frac{\pi}{4}\right)\} = \pi e^{-j\pi/4}\delta(\omega + 30\pi) + \pi e^{j\pi/4}\delta(\omega - 30\pi)]$$

Next, we perform a variable change by setting ($\omega = 2\pi f$). Considering the necessity of normalizing the function, we omit the (π) coefficient:

$$[\mathcal{F}_N\{\cos\left(30\pi t + \frac{\pi}{4}\right)\} = e^{-j\pi/4}\delta(f + 15) + e^{j\pi/4}\delta(f - 15)]$$

As observed, the theoretical calculations align with the obtained values.

- ***Part 2:***

2.1) we create a map set:

```matlab
%% 2.1 creating a mapset

Alphabet='abcdefghijklmnopqrstuvwxyz .,!;"';

num_alphabet=length(Alphabet);
mapset=cell(2,num_alphabet);
for i=1:num_alphabet
    mapset{1,i}=Alphabet(i);
    mapset{2,i}=dec2bin(i-1,5);
end
char_bin_len = length(mapset{2, 1});

```

2.2) This function, coding_freq, encodes a binary message into a frequency-shift keying (FSK) signal. It takes two parameters: bin_msg represents the binary message to be encoded, and bit_rate determines the number of bits used to represent each frequency symbol. The function initializes parameters such as the sampling frequency (fs), time vector (t), and frequency vector (f). It then iterates through the binary message, encoding each group of bits into a corresponding sinusoidal signal with a frequency determined by the binary value. The frequency is computed based on the partitioning of the frequency spectrum into regions, and each encoded signal is concatenated to create the final FSK signal. The result, signal, represents the FSK-modulated waveform of the input binary message.

```matlab
function signal = coding_freq(bin_msg, bit_rate)
    fs = 100;
    N = 100;
    t_start = 0;
    Ts = 1/fs;
    T_end = 1;
    t = t_start:Ts:T_end;
    f=0:fs/N:(fs/2) - fs/N;
    partition = floor(length(f) / (2^bit_rate));
    middle_part = ceil(length(f) / ((2^bit_rate) * 2));
    coded_signal = [];

    for i = 1:bit_rate:length(bin_msg)
        frequency = partition * bin2dec(bin_msg(i:i+bit_rate-1)) + middle_part;
        y = sin(2 * pi * frequency * t);
        coded_signal = [coded_signal, y];
    end

    signal = coded_signal;
end
```

2.3) Here is the result for the signal with 1 and 5 bitrate per second:

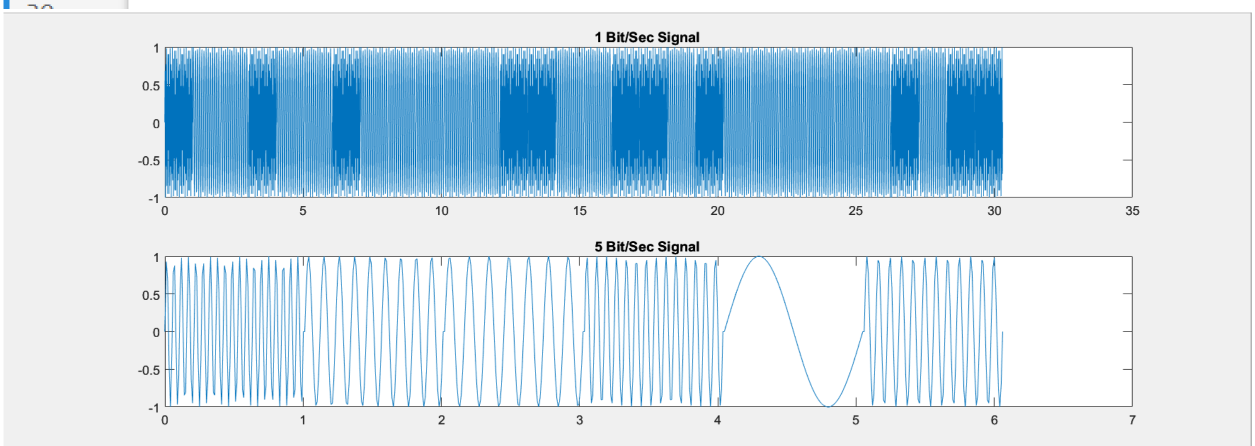
```
%% 2.2/2.3 coding a massage

fs = 100;
msg = 'signal';
index=[];
for i=1:length(msg)
    ch=msg(i);
    index=[index, find(strcmp(ch,mapset(1,:))==1)];
end
bin_msg=cell2mat(mapset(2,index));

figure

bit_rate=1;
subplot(2, 1, 1);
coded_signal = coding_freq(bin_msg, bit_rate);
t = linspace(0, length(coded_signal) / fs, length(coded_signal));
plot(t, coded_signal);
title([num2str(bit_rate),' Bit/Sec Signal ']);

bit_rate=5;
subplot(2, 1, 2);
coded_signal = coding_freq(bin_msg, bit_rate);
t = linspace(0, length(coded_signal) / fs, length(coded_signal));
plot(t, coded_signal);
title([num2str(bit_rate),' Bit/Sec Signal ']);
```



2.4) This function decoding_freq that takes a signal and a bit rate as input parameters and performs frequency domain decoding of the signal. The function divides the frequency spectrum into intervals based on the specified bit rate, and then, for each chunk of the input signal, it computes the dominant frequency using FFT. It compares the identified frequency with predefined thresholds to determine the corresponding binary representation, and appends the decoded bits to a binary message. The function utilizes FFT and frequency domain analysis to convert frequency information into binary data, providing a method for decoding signals in a communication system where different frequencies represent distinct information. The code assumes a sampling frequency fs of 100 Hz and a signal duration of 1 second. The resulting binary message represents the decoded information from the input signal.

```
function binary_msg=decoding_freq(signal,bit_rate)
    fs=100;
    N=100;
    Tend=1;
    chunk = round(Tend * fs);
    binary_msg='';
    Thresholds=[];
    f=0:fs/N:(fs/2) - fs/N;
    partition=length(f)/(2^bit_rate);

    for i=0:1:(2^bit_rate-1)
        Thresholds=[Thresholds partition*i];

    end

    Thresholds=[Thresholds N/2];

    for i = 1:chunk:(length(signal) - chunk)
        sig_part = signal(i:i+chunk-1);
```

```
    for i = 1:chunk:(length(signal) - chunk)
        sig_part = signal(i:i+chunk-1);
        y=fftshift(fft(sig_part));
        F=y/max(abs(y));
        f=-(fs/2):(fs/N):(fs/2)-(fs/N);
        row = find(abs(F) == max(abs(F)));
        m=f(row);
        found_freq= m(m>0);
        decoded_bits = 0;
        for i=(2^bit_rate+1):-1:1
            if Thresholds(i) <= found_freq
                decoded_bits = i -1;
                break;

            elseif (i == 1)
                decoded_bits = 0;
            end
        end
        binary_msg=strcat(binary_msg,dec2bin(decoded_bits,bit_rate));
    end
end
```

Here are the result of the function:

```
the encoded massage (with the bitrate =1)is :signal
the encoded massage (with the bitrate =5)is :signal
```

2.5) Yes, we were able to extract the message from the signal with added noise. The results are visible below.

```
the encoded massage (with the bitrate =1 and noise =0.01)is :signal
the encoded massage (with the bitrate =5 and noise =0.01)is :signal
```

2.6-7) with the below code we systematically iterated through different noise amplitudes (noise_amp) and two specific bit rates (1 and 5). For each combination, we generated a coded signal using the coding_freq function, introduced Gaussian noise, and subsequently decoded the noisy signal using decoding_freq. The decoded binary signal was then converted to a string using binary_to_string. We printed and examined the decoded message, checking for a match with a reference string ('signal'). If a match was found, we updated the max_noise_var array to record the maximum noise variance permitting successful decoding for each bit rate. The final results, indicating the maximum noise variance for successful decoding, were displayed after the iterations.

```
for j = 1:length(bit_rates)
for i = 1: length(noise_amp)


    bit_rate = bit_rates(j);
    coded_signal = coding_freq(bin_msg, bit_rates(j));
    noisy_signal = coded_signal + noise_amp(i) * randn(size(coded_signal));
    binary_decoded_signal = decoding_freq(noisy_signal, bit_rate);
    decoded_signal = binary_to_string(binary_decoded_signal, mapset);
    disp(['the encoded massage (with the bitrate =',num2str(bit_rate),' and noise =',num2str(noise_amp(i)),')is :', decoded_signal])
    if ~strcmp(decoded_signal,'signal')
        max_noise_var(j) = max(max_noise_var(j),noise_amp(i));
        break
    end
end
end
max_noise_var
```

The maximum noise variance allowing successful decoding was 0.91 for a bit rate of 1 and 1.14 for a bit rate of 5. Therefore, the decoding process demonstrated greater resistance to noise for a bit rate of 5 compared to a bit rate of 1. But the result were not the same with introduction.

```
max_noise_var =

    0.9100    1.1400
```

2.9) If we increase the sampling rate without increasing the bandwidth, we can still enhance the resistance to noise. This concept, known as oversampling, involves acquiring more samples per second, allowing for the transmission of more information within a given time interval. While the bandwidth remains constant, the increased number of samples per second contributes to improved resistance to noise and the potential for higher data transmission speeds, especially applicable in digital communication and internet scenarios.