Communication System Lab 6

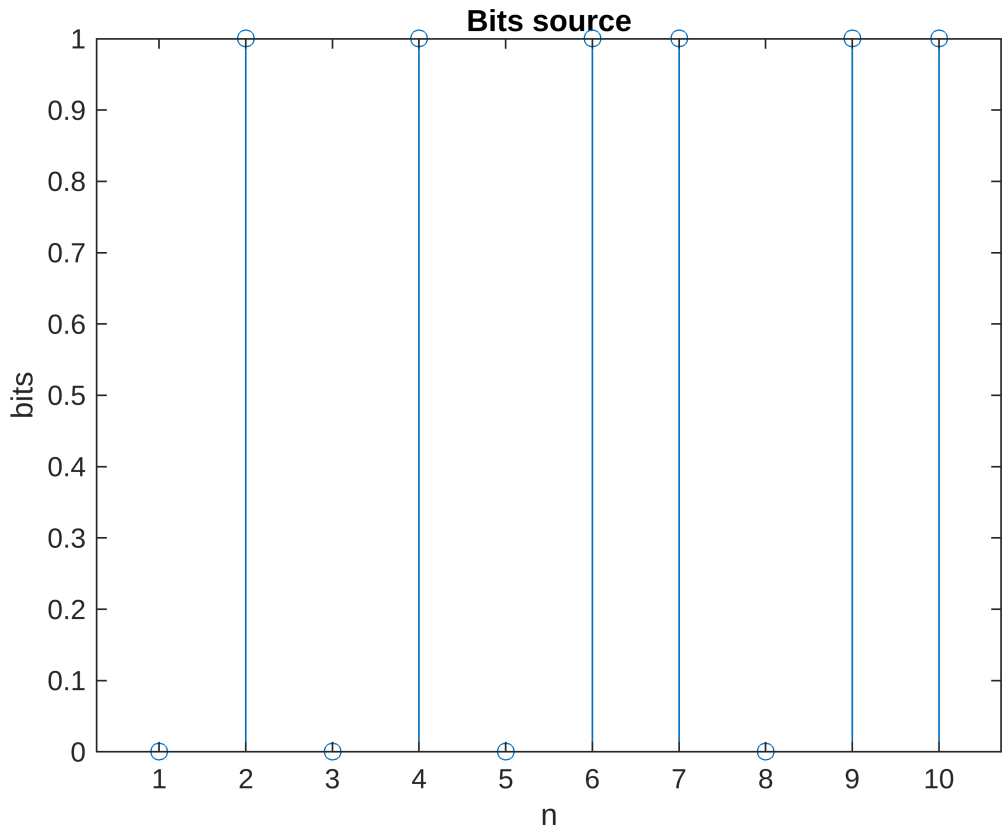
Passband Communication

By: Saurabh Kumar (SC22B146)

1 Basic passband modulation - BPSK

1. Generation of bits: The source is assumed to produce a stream of bits. The bits are usually modelled as an independent and identically distributed random process with the probability of a bit being 1 being 0.5. Generate a random sequence of bits of length N (input) satisfying this property.

```
N = 10;

% Bits generation
source = rand(1, N) > 0.5;

figure;
stem(source);
title("Bits source");
xlabel("n");
ylabel("bits");
```

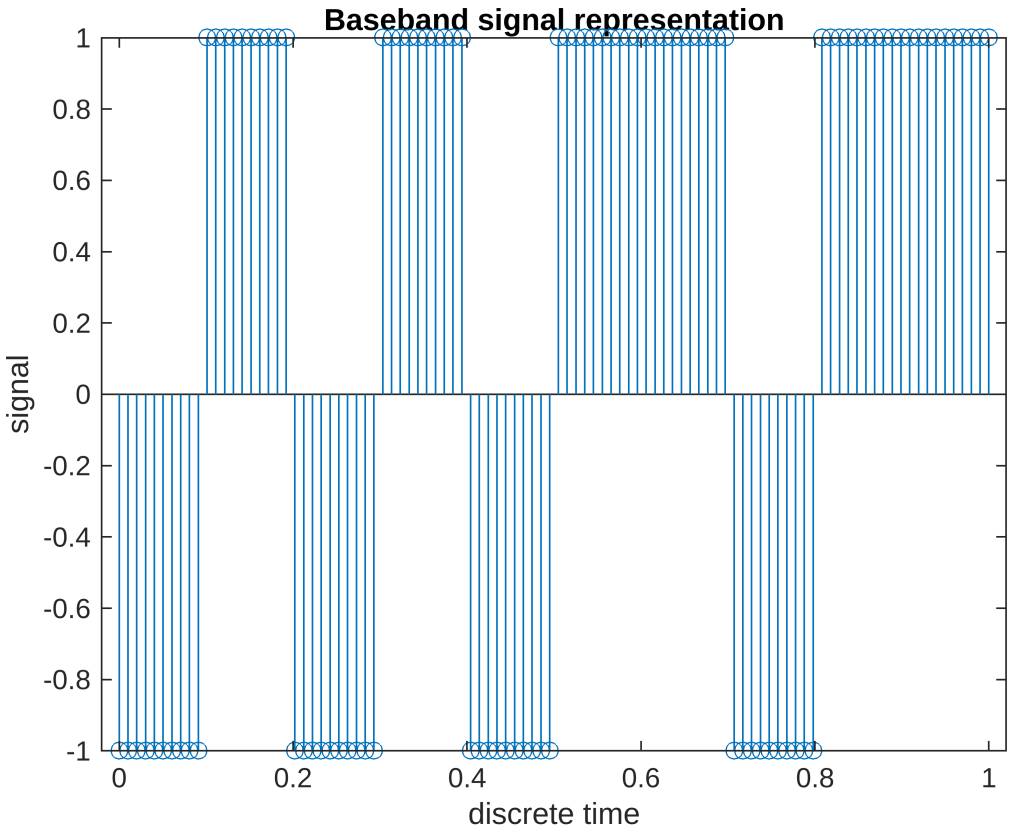



```
disp(['Bits are: ', num2str(source)]);
```

```
Bits are: 0  1  0  1  0  1  1  0  1  1
```

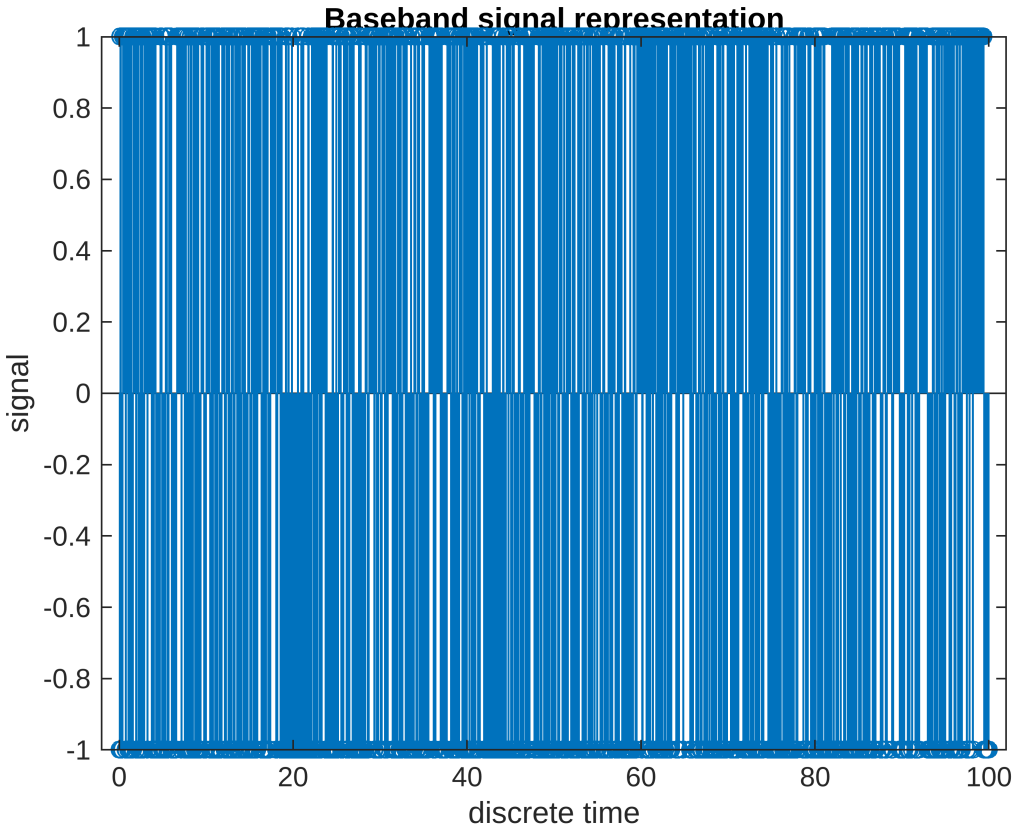
2. The above bit sequence is then converted to a baseband signal. The baseband signal is obtained by converting 0 and 1 into appropriate pulses of duration T = 1s. For a bit sequence with N = 10, obtain a baseband signal with 0 represented using a level of −1 and 1 using a level of 1. Note that all continuous time signals have to be represented by their sampled counterparts. Clearly state the sampling rate that you have used. Plot the sampled version of the continuous time signal that you have obtained.

```
T = 0.1;
fs = 100;
n = T*fs;

% Signal generation
signal = zeros(1,n*N);
for i=1:(length(source))
    signal((i-1)*n+1:(i)*n) = repelem(source(i),n);
end
signal(signal==0) = -1;

figure;
t = linspace(0,N*T,N*fs*T);
stem(t,signal);
title("Baseband signal representation");
xlabel("discrete time");
ylabel("signal");
```

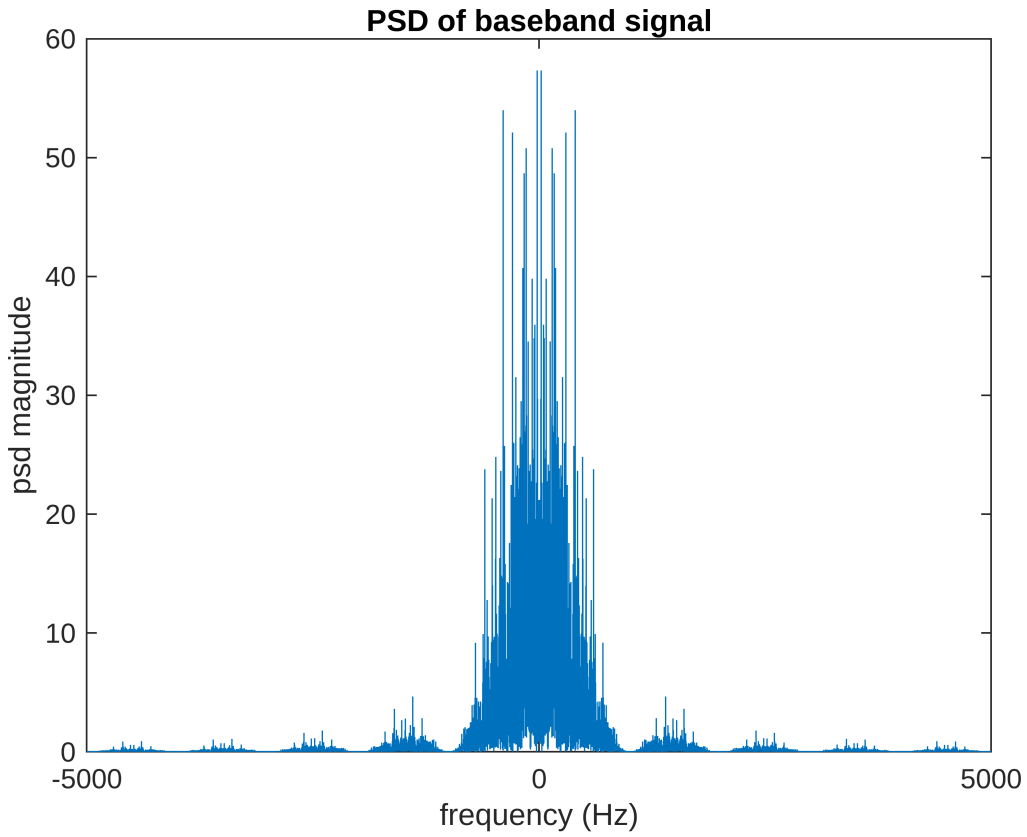


3. Plot the power spectrum of the baseband signal for N = 1000. Note that the power spectrum should be plotted with respect to the continuous time frequency (Hint: carefully think about energy vs. power).

```
N = 1000;
T = 0.1;
fs = 100;
n = T*fs;

% Bits and Signal generation
source = rand(1, N) > 0.5;

signal = zeros(1,n*N);
for i=1:(length(source))
    signal((i-1)*n+1:(i)*n) = repelem(source(i),n);
end
signal(signal==0) = -1;

figure;
t = linspace(0,N*T,N*fs*T);
stem(t,signal);
title("Baseband signal representation");
xlabel("discrete time");
ylabel("signal");
```

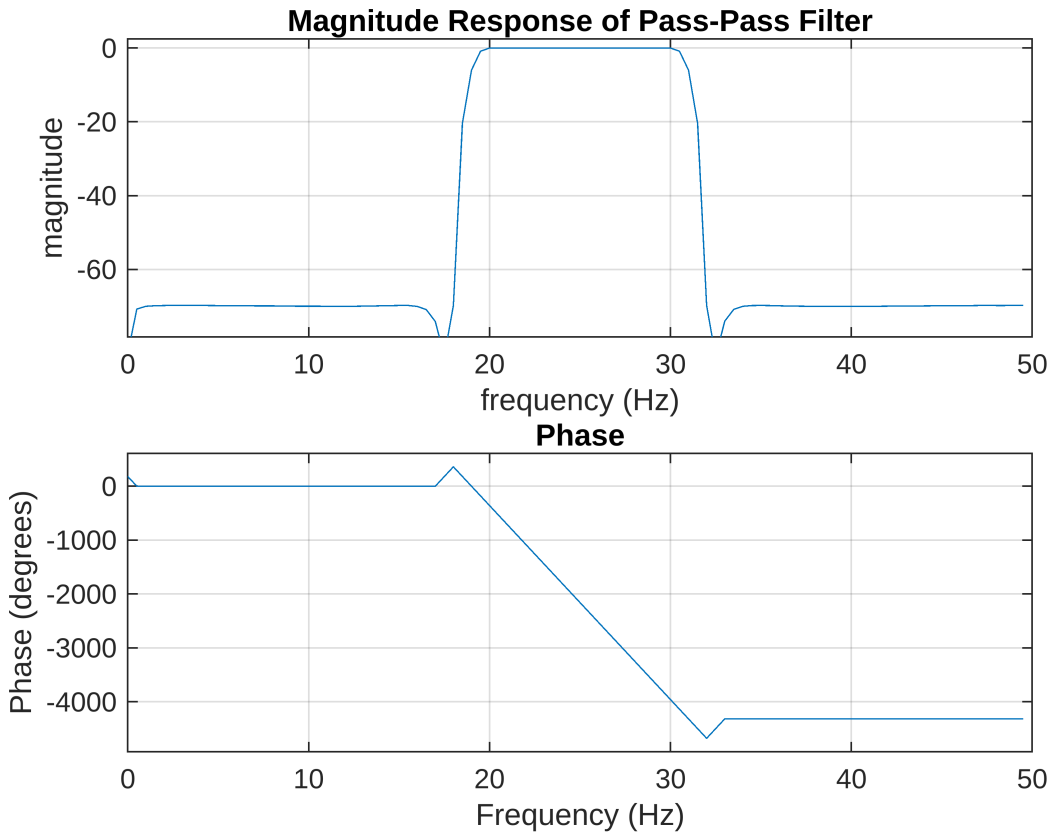


```
% PSD of baseband signal
psd = fftshift((fft(signal)).^2)/length(signal);
figure;
f = -(length(psd)/2):(length(psd)/2-1);
plot(f,abs(psd));
title("PSD of baseband signal");
xlabel("frequency (Hz)");
ylabel("psd magnitude");
```



4. For modelling a passband channel use any FIR filter design technique to obtain a filter response that corresponds to a channel with passband centered at 25Hz, one-sided bandwidth of 10 Hz, and a gain of g in the passband. Plot the magnitude spectrum of the channel that you are using - clearly labelling the axes for g = 1.

```
fs = 100;

% Passband channel
channel = firpm(200, [0, 18, 20, 30, 32, fs/2]/(fs/2),[0, 0, 1, 1, 0, 0]);

figure;
freqz(channel, 1, 100, fs);
title('Magnitude Response of Pass-Pass Filter');
xlabel('frequency (Hz)');
ylabel('magnitude');
```

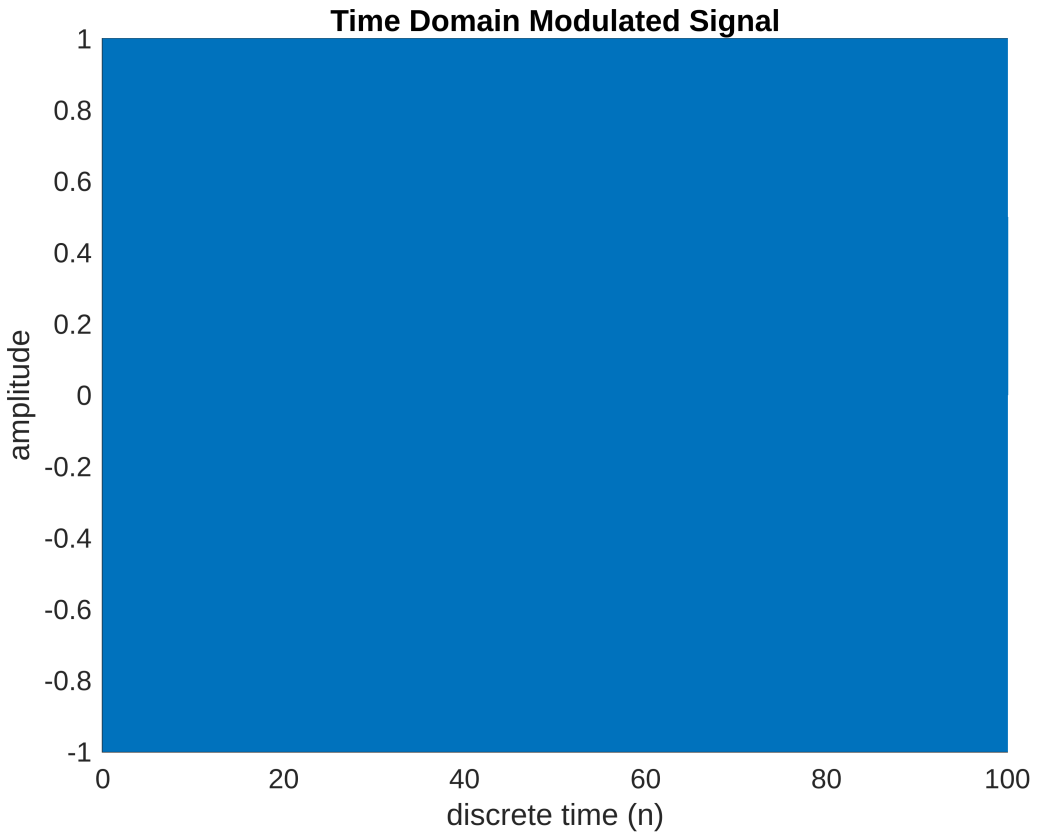


5. Note that the baseband signal that you have generated cannot be sent through the passband channel directly. So modulate the baseband signal using a carrier of frequency 25 Hz.

```
fc = 25;
t = (0:length(signal)-1)/fs;

% Modulated signal
mod_signal = cos(2*pi*fc*t).*signal;
```

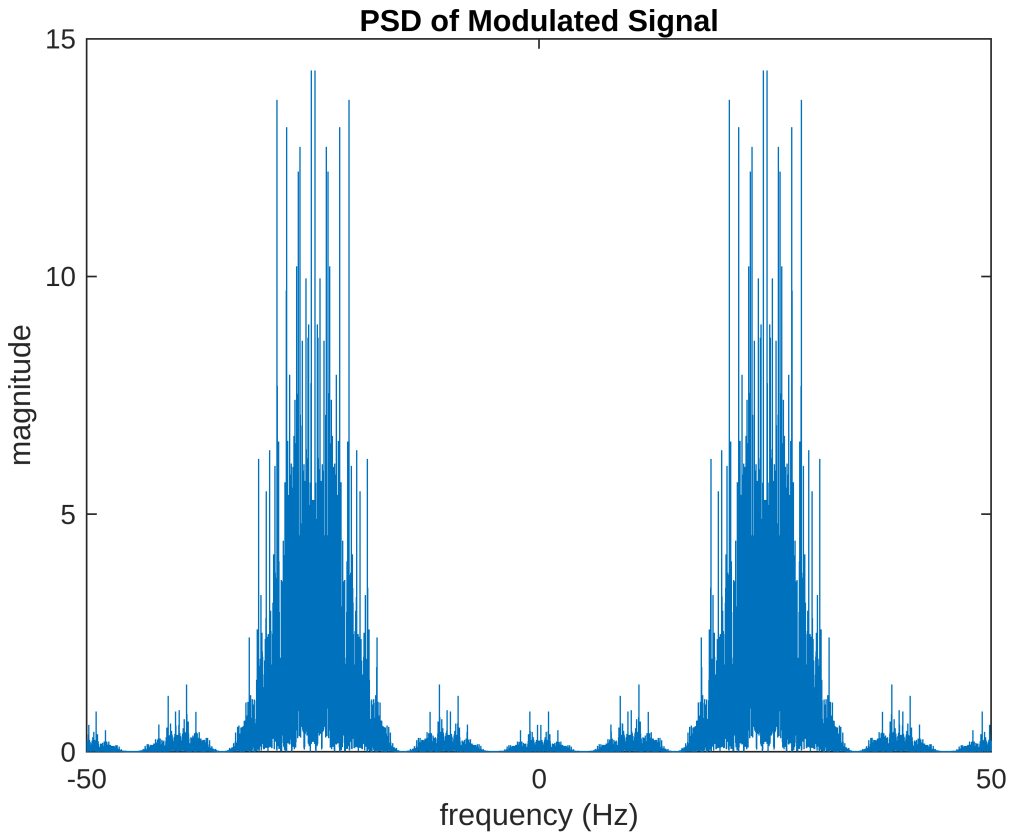
6. Plot the modulated signal in time domain - clearly labelling the axis.

```
figure;
t = linspace(0,N*T,N*fs*T);
plot(t,mod_signal);
title('Time Domain Modulated Signal');
xlabel('discrete time (n)');
ylabel('amplitude');
```



7. Plot the power spectral density of the modulated signal. Compare the PSD of the modulated signal with that of the baseband signal. State your observations - what are the similarities and differences between the two PSDs?

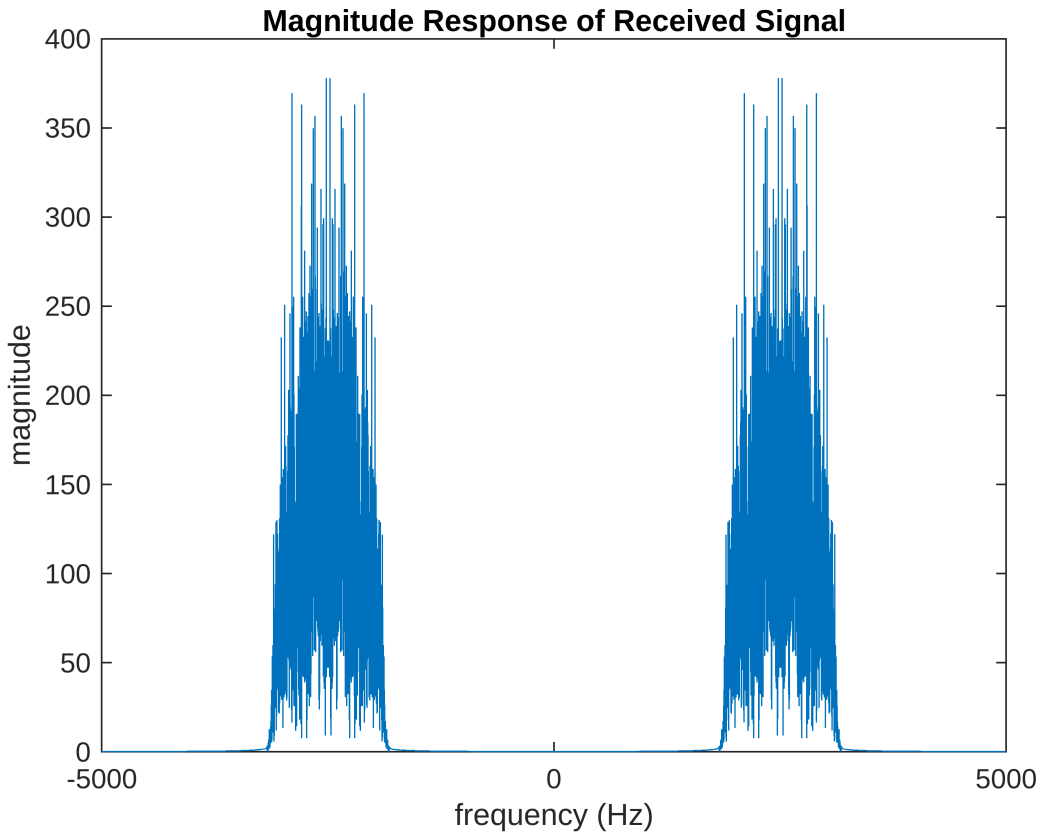
```
% PSD of modulated signal
len_mod_signal = length(mod_signal);
psd = abs((fft(mod_signal)).^2)/len_mod_signal;

figure;
f = (-len_mod_signal/2:len_mod_signal/2-1)*(fs/len_mod_signal);
plot(f,psd);
title("PSD of Modulated Signal");
xlabel("frequency (Hz)");
ylabel("magnitude");
```



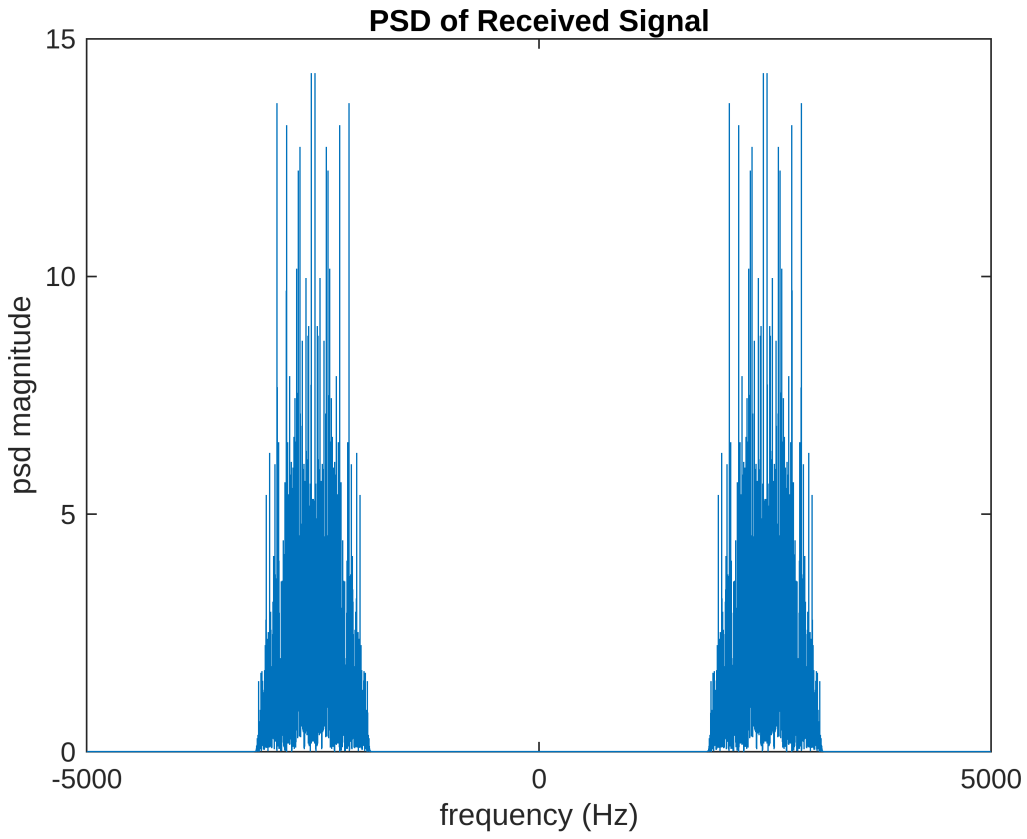
Inference: The PSD of the baseband signal is centered around 0 Hz, while the PSD of the modulated signal is shifted to ±25 (carrier frequency); both share the same shape but differ in spectral location.

8. Simulate sending the passband signal through the above channel. Plot the output signal from the channel as well as its PSD.

```
% Received signal
receive_signal = conv(mod_signal,channel,'same');
receive_signal_freq = abs(fft(receive_signal));

figure;
f = -(length(receive_signal_freq)/2):(length(receive_signal_freq)/2-1);
plot(f,receive_signal_freq);
title('Magnitude Response of Received Signal');
xlabel('frequency (Hz)');
ylabel('magnitude');
```

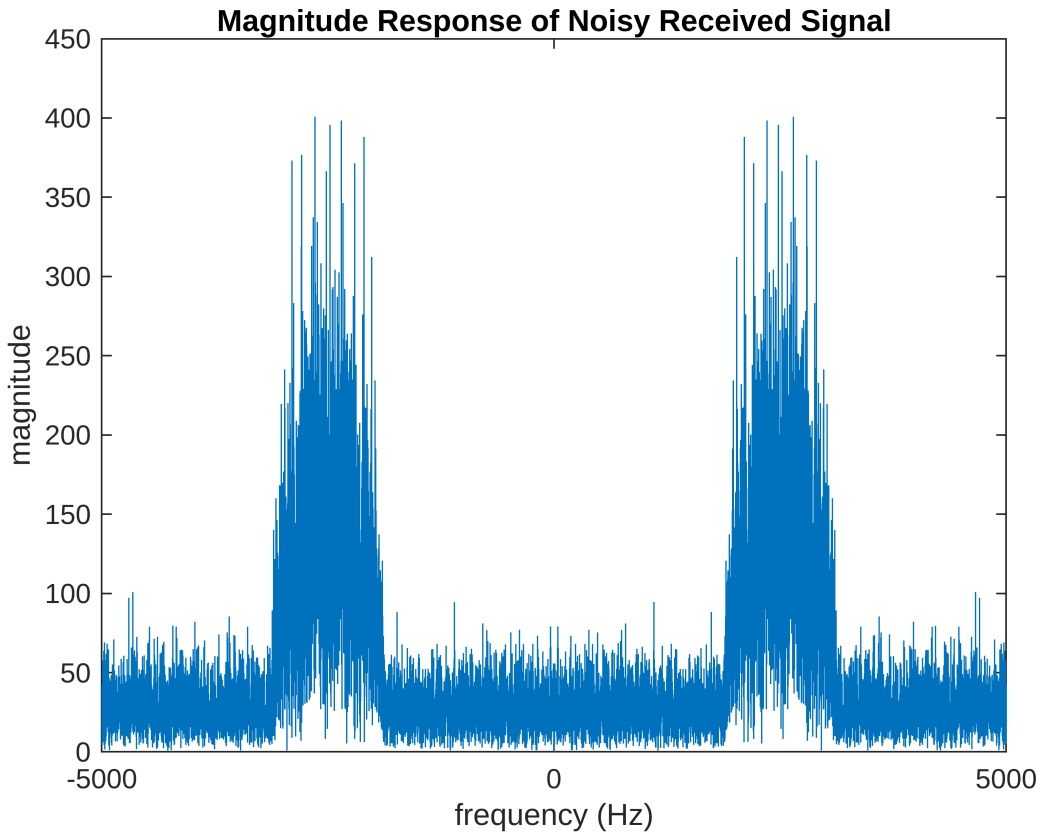



```
% PSD of received signal
psd = fftshift((fft(receive_signal)).^2)/length(receive_signal);
figure;
f = -(length(psd)/2):(length(psd)/2-1);
plot(f,abs(psd));
title("PSD of Received Signal");
xlabel("frequency (Hz)");
ylabel("psd magnitude");
```



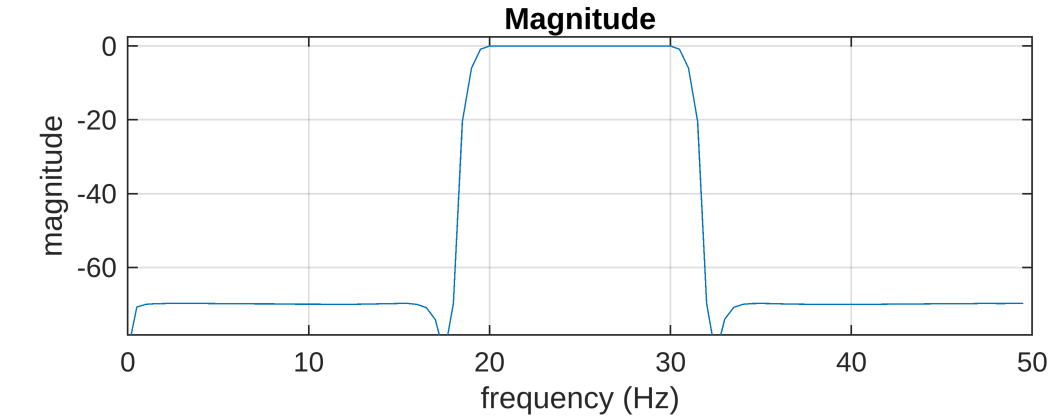
9. Now implement the additive noise model for the channel so that the channel introduces additive white Gaussian noise with variance σ2 . For this set of tasks use σ2 = 0.1.

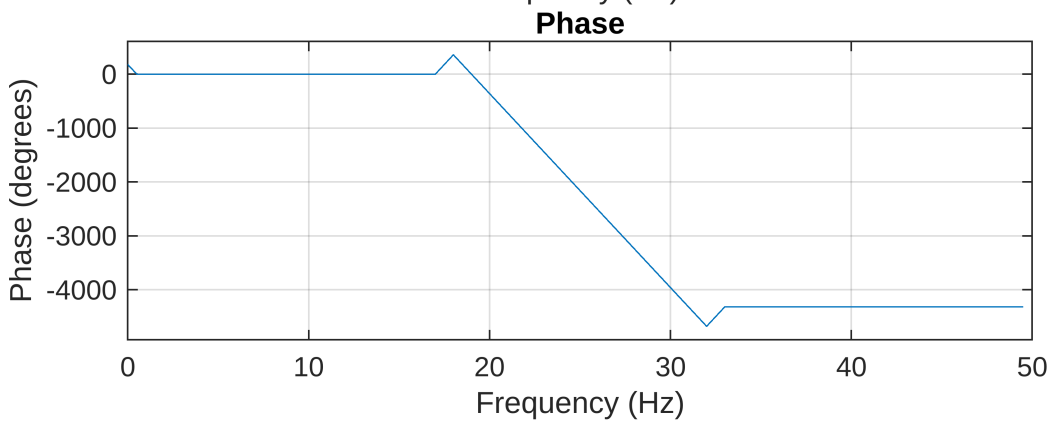
```
% Noise
noise_var = 0.1;
noise_stddev = sqrt(noise_var);
noise = noise_stddev * randn(1,length(receive_signal));

% Received Signal
receive_signal_noisy = receive_signal + noise;
receive_signal_noisy_freq = abs(fft(receive_signal_noisy));

figure;
f = -(length(receive_signal_noisy_freq)/2):
(length(receive_signal_noisy_freq)/2-1);
plot(f,receive_signal_noisy_freq);
title('Magnitude Response of Noisy Received Signal');
xlabel('frequency (Hz)');
ylabel('magnitude');
```



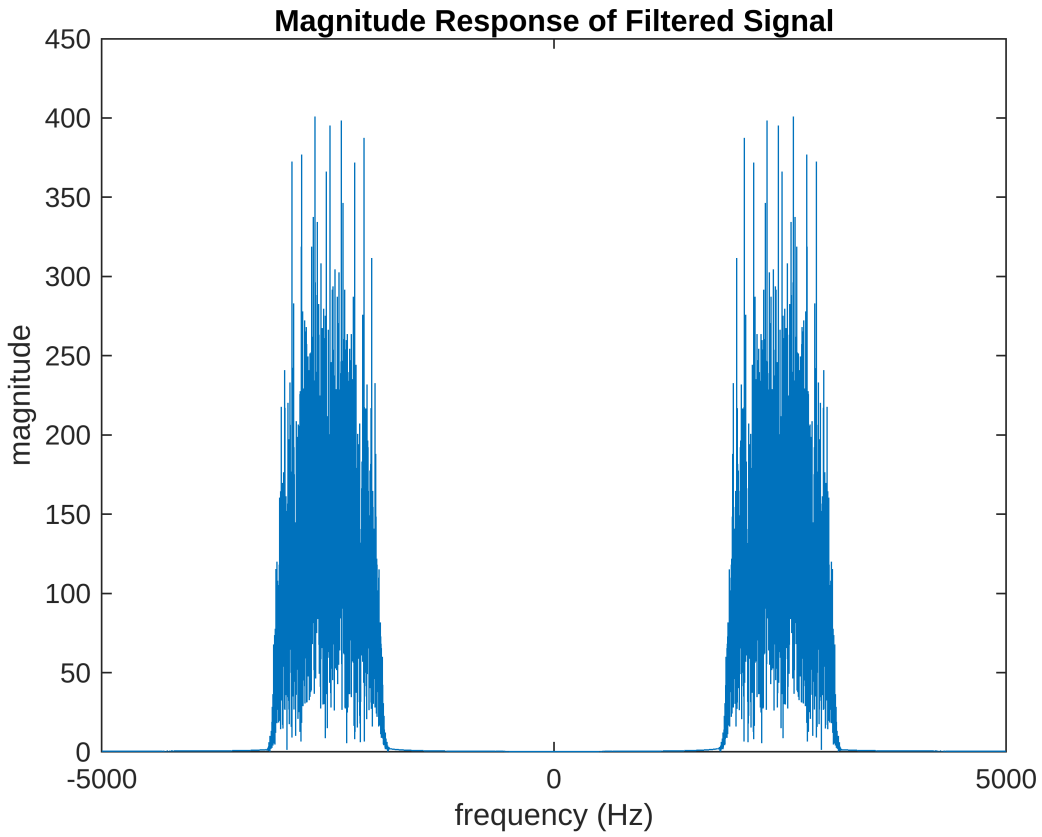
10. At the output of the ideal channel, implement a bandpass receive filter. Plot the magnitude response of the filter that you have implemented. What are the specifications of the passband and stopband for this filter? Why did you choose those specifications?

```
fs = 100;

% Bandpass receive filter
receiver_filter = firpm(200, [0, 18, 20, 30, 32, fs/2]/(fs/2),[0, 0, 1, 1,
0, 0]);

figure;
freqz(receiver_filter, 1, 100, fs);
xlabel('frequency (Hz)');
ylabel('magnitude');
```






```
% Filtered signal
filtered_signal = conv(receive_signal_noisy,receiver_filter,'same');
filtered_signal_freq = abs(fft(filtered_signal));

figure;
f = -(length(filtered_signal_freq)/2):(length(filtered_signal_freq)/2-1);
plot(f,filtered_signal_freq);
title('Magnitude Response of Filtered Signal');
xlabel('frequency (Hz)');
ylabel('magnitude');
```



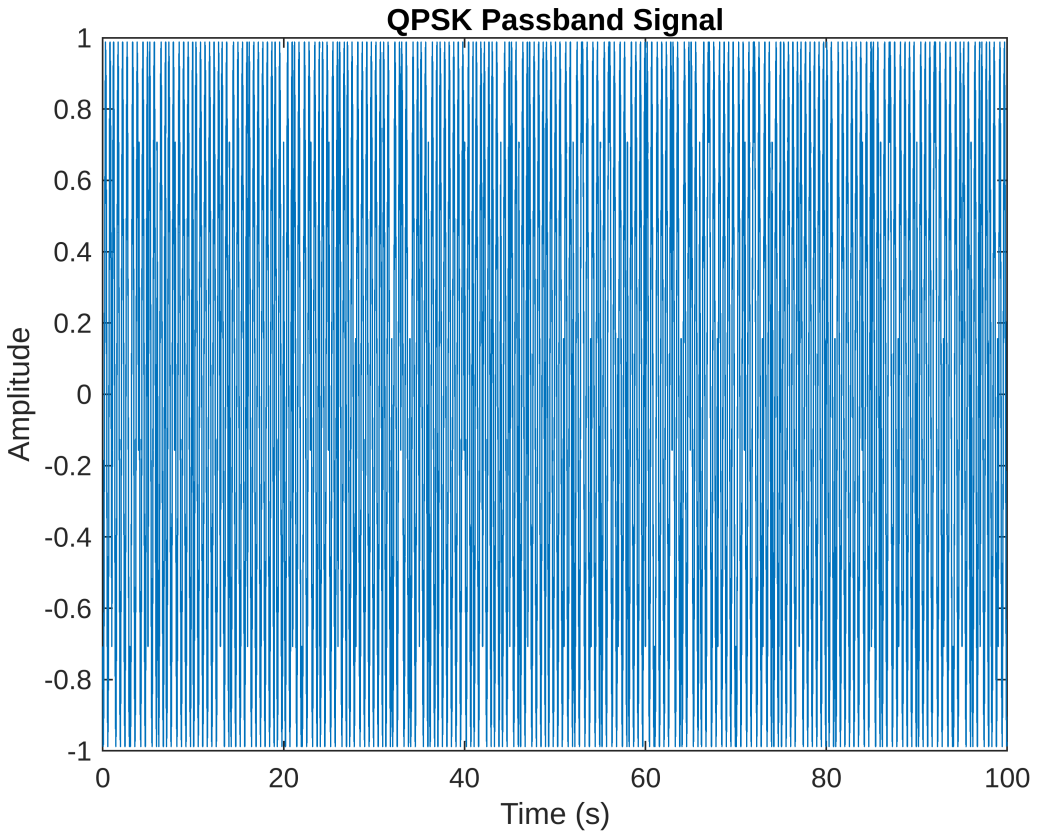
Inference: Passband: 20–30 Hz, Stopbands: 0–18 Hz and 32–50 Hz; chosen to isolate a 25 Hz carrier modulated signal while rejecting out-of-band noise.

2 General passband modulation

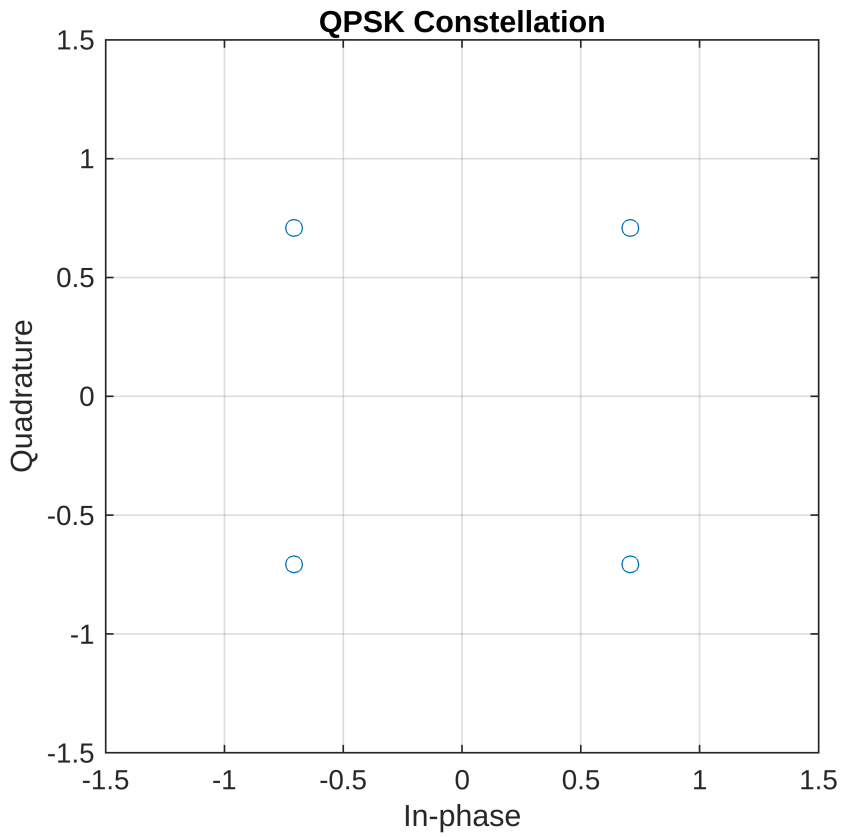
1. For a sequence of 200 bits obtain the waveform corresponding to QPSK. The symbol time can be chosen to be 1 second. Record the bit rate. Plot the PSD of the QPSK waveform and record the null to null bandwidth. Plot the signal constellation for this transmission scheme.

```
N = 200;
fs = 20; % Samples per symbol
symbol_time = 1;
fc = 2;
Ts = symbol_time / fs; % Time step
num_symbols = N / 2;

% Bits and symbols generation
bits = randi([0 1], 1, N);
bit_pairs = reshape(bits, 2, [])'; % Each row is 2 bits

% Map bits to QPSK using Gray Code
symbols = zeros(1, num_symbols);
for k = 1:num_symbols
    b1 = bit_pairs(k,1);
```

```
    b2 = bit_pairs(k,2);

    if     b1==0 && b2==0, symbols(k) = (1 + 1j) / sqrt(2);
    elseif b1==0 && b2==1, symbols(k) = (-1 + 1j) / sqrt(2);
    elseif b1==1 && b2==1, symbols(k) = (-1 - 1j) / sqrt(2);
    elseif b1==1 && b2==0, symbols(k) = (1 - 1j) / sqrt(2);
    end
end

% Time vector and waveform
t = 0 : Ts : (num_symbols * symbol_time - Ts);
s = zeros(1, length(t));

for k = 1:num_symbols
    idx = (k-1)*fs + 1 : k*fs;
    I = real(symbols(k));
    Q = imag(symbols(k));
    s(idx) = I*cos(2*pi*fc*t(idx)) - Q*sin(2*pi*fc*t(idx));
end

% Time Domain Signal
figure;
plot(t, s);
xlabel('Time (s)'); ylabel('Amplitude');
title('QPSK Passband Signal');
```



```
% Constellation
figure;
plot(real(symbols), imag(symbols), 'o');
xlabel('In-phase'); ylabel('Quadrature');
title('QPSK Constellation');
grid on; axis equal;
axis([-1.5 1.5 -1.5 1.5]);
```




```
% Bit Rate
bit_rate = N / (num_symbols * symbol_time);
disp(['Bit rate = ', num2str(bit_rate), ' bps']);
```

```
Bit rate = 2 bps
```

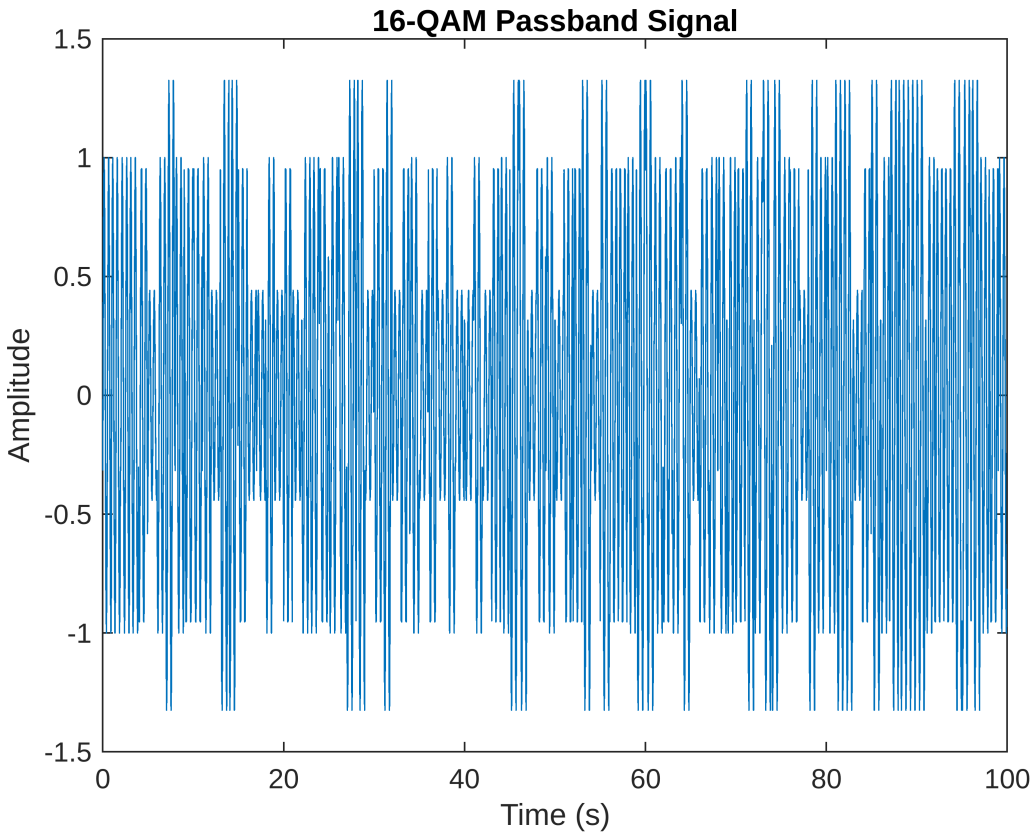
```
% Null-to-Null Bandwidth
disp(['Null-to-null bandwidth = ', num2str(2 / symbol_time), ' Hz']);
```

```
Null-to-null bandwidth = 2 Hz
```

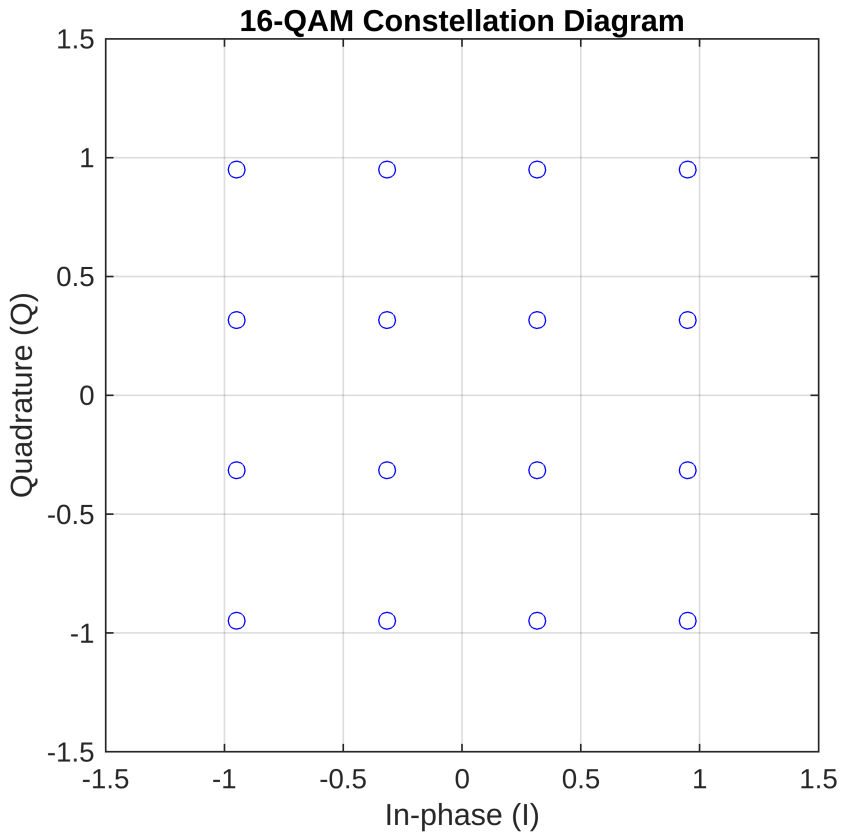
2. For a sequence of 400 bits obtain the waveform corresponding to 16-QAM. Again record the bit rate and the null to null bandwidth. Plot the signal constellation for this transmission scheme.

```
N = 400;
M = 16;
k = log2(M); % bits per symbol = 4
symbol_count = N / k;
```

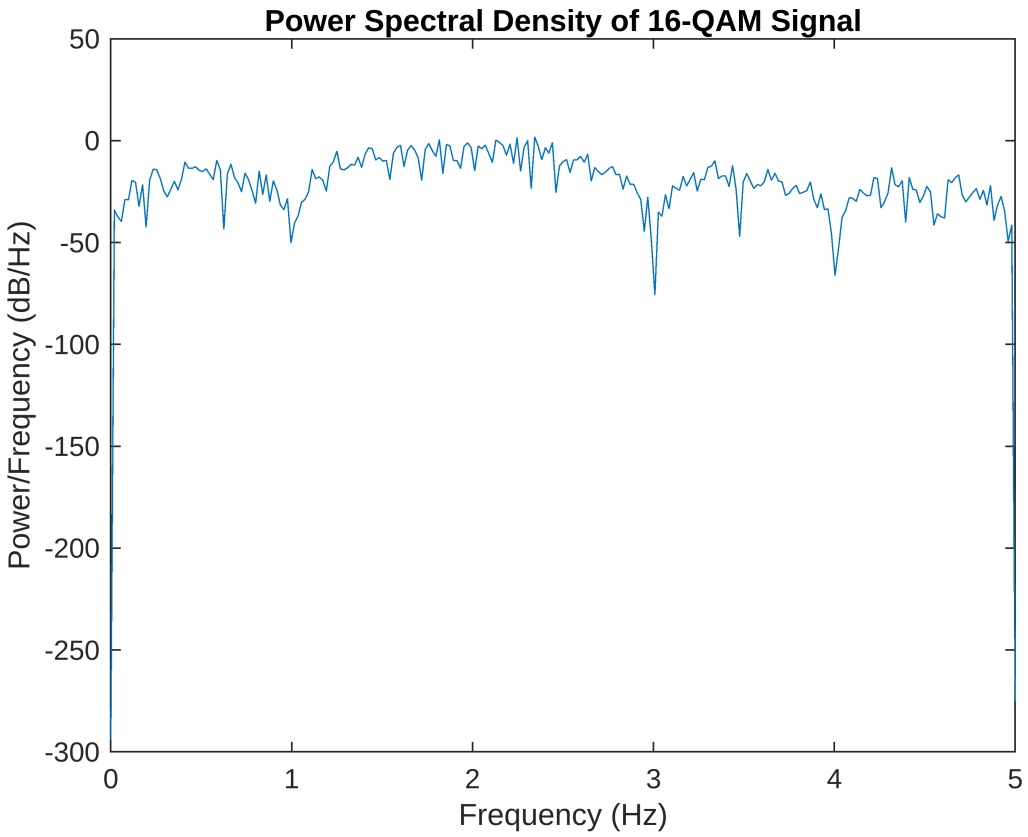
```
fs = 20; % samples per symbol
fc = 2;
symbol_time = 1;
Ts = symbol_time / fs;

bits = randi([0 1], 1, N);

% Gray-coded bit-to-symbol mapping
I_levels = [-3, -1, +3, +1];
Q_levels = [-3, -1, +3, +1];

% Apply Gray coding to symbol map
gray_map = containers.Map(...
    {'0000','0001','0011','0010','0110','0111','0101','0100', ...
     '1100','1101','1111','1110','1010','1011','1001','1000'}, ...
    num2cell([1:16]));

% Symbol mapping (I,Q)
constellation = zeros(1, symbol_count);
for i = 1:symbol_count
    group = bits((i-1)*k + 1:i*k);
    key = num2str(group);
    key = strrep(key, ' ', '');

    idx = gray_map(key);

    row = ceil(idx / 4); % Q level index
    col = mod(idx-1, 4) + 1; % I level index
    I = I_levels(col);
    Q = Q_levels(row);

    constellation(i) = (I + 1j*Q) / sqrt(10);
end

% Waveform
t = 0:Ts:(symbol_count * symbol_time - Ts);
s = zeros(1, length(t));
for i = 1:symbol_count
    idx = (i-1)*fs + 1 : i*fs;
    I = real(constellation(i));
    Q = imag(constellation(i));
    s(idx) = I*cos(2*pi*fc*t(idx)) - Q*sin(2*pi*fc*t(idx));
end

% Time Domain Signal
figure;
plot(t, s);
xlabel('Time (s)');
ylabel('Amplitude');
title('16-QAM Passband Signal');
```




```
% Signal Constellation
figure;
plot(real(constellation), imag(constellation), 'bo');
xlabel('In-phase (I)');
ylabel('Quadrature (Q)');
title('16-QAM Constellation Diagram');
grid on; axis equal;
axis([-1.5 1.5 -1.5 1.5]);
```




```
% PSD
figure;
nfft = 1024;
[psd_vals, f] = periodogram(s, [], nfft, 1/Ts);
plot(f, 10*log10(psd_vals));
xlabel('Frequency (Hz)');
ylabel('Power/Frequency (dB/Hz)');
title('Power Spectral Density of 16-QAM Signal');
xlim([0, 5]);
```




```
% Bit Rate and Null-to-Null Bandwidth
bit_rate = N / (symbol_count * symbol_time);
null_to_null_bw = 2 / symbol_time;
disp(['Bit rate = ', num2str(bit_rate), ' bps']);
```

```
Bit rate = 4 bps
```

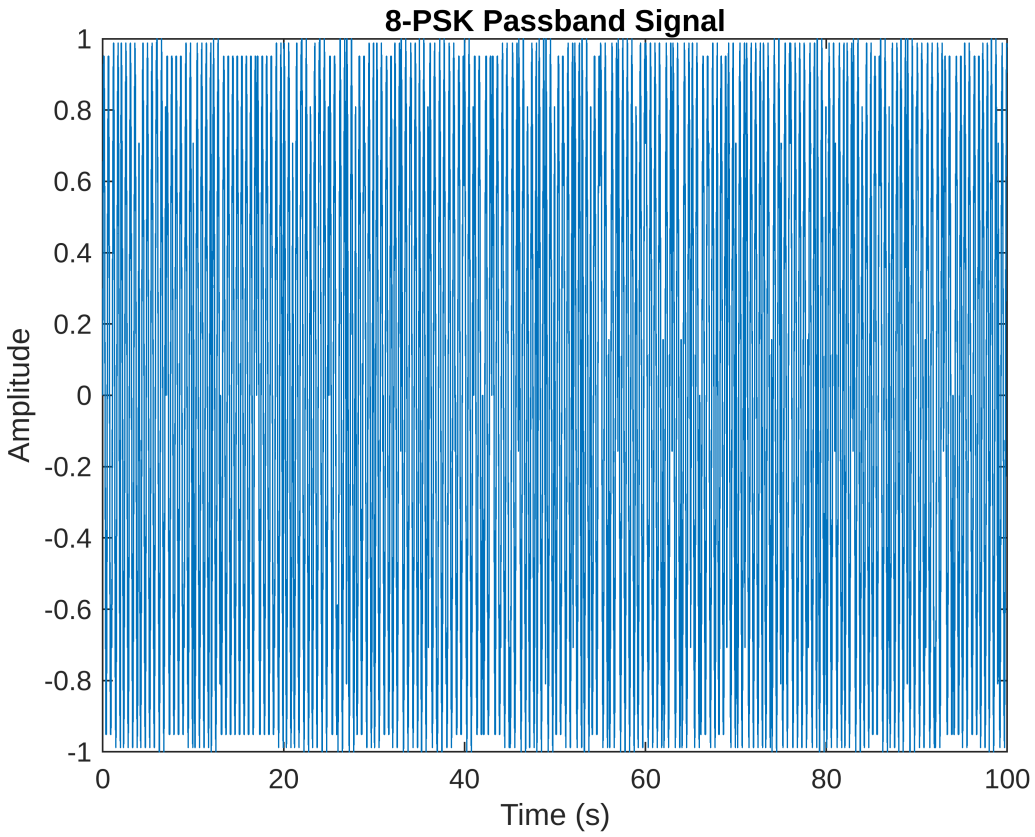
```
disp(['Null-to-null Bandwidth = ', num2str(null_to_null_bw), ' Hz']);
```

```
Null-to-null Bandwidth = 2 Hz
```

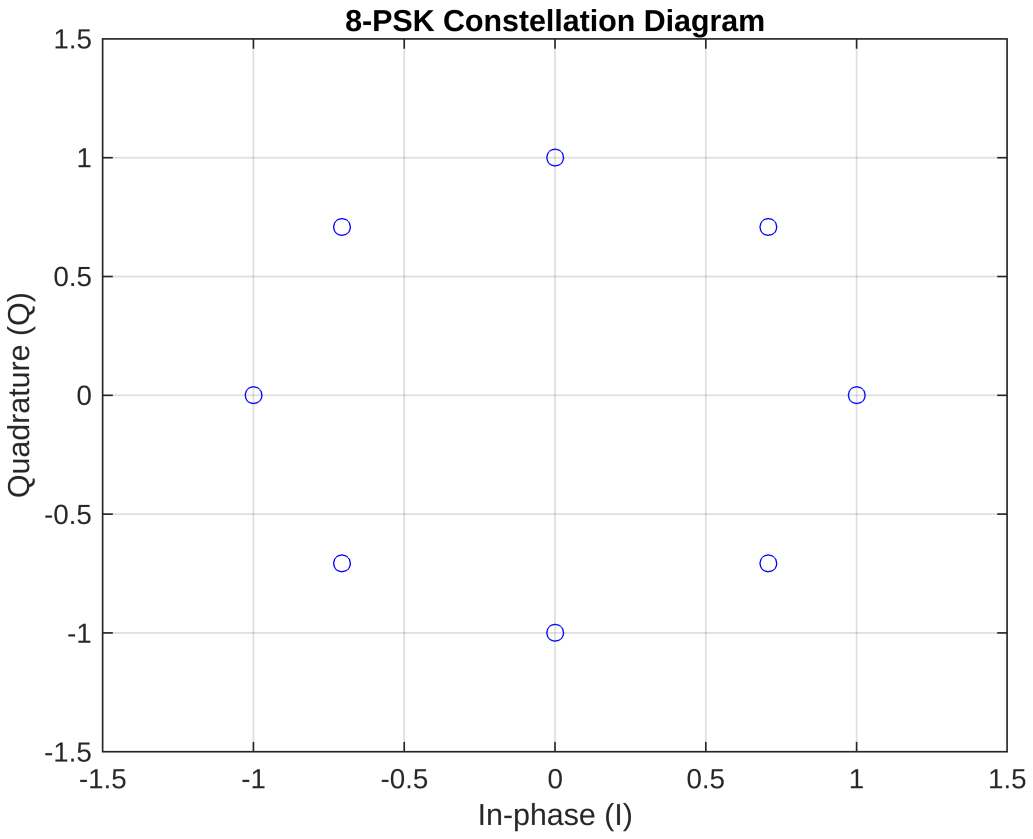
3. For a sequence of 300 bits obtain the waveform corresponding to 8-PSK. Again record the bit rate and the null to null bandwidth. Plot the signal constellation for this transmission scheme.

```
N = 300;
k = 3; % Bits per symbol
M = 2^k;
symbol_count = N / k;
fs = 20; % Samples per symbol
symbol_time = 1;
Ts = symbol_time / fs; % Sampling interval
fc = 2;

bits = randi([0 1], 1, N);
```

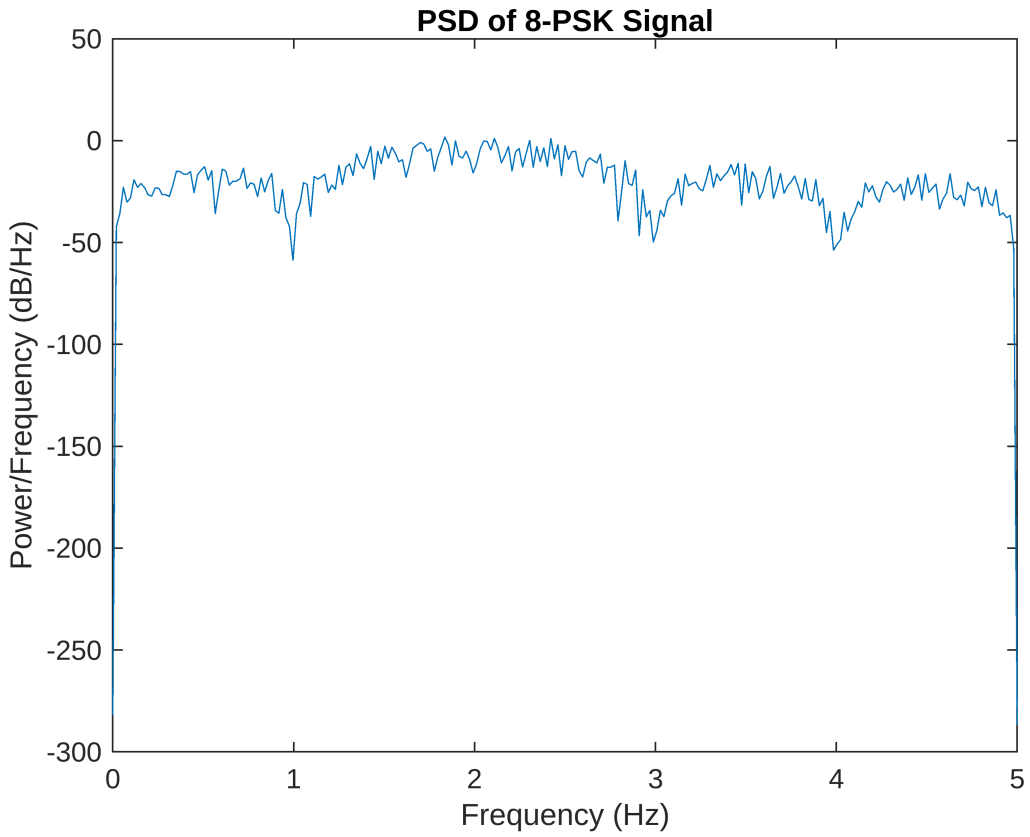
```
% Bits to symbols conversion
symbols = zeros(1, symbol_count);
for i = 1:symbol_count
    group = bits((i-1)*k + 1 : i*k);
    symbols(i) = bin2dec(num2str(group));
end

% Mapping to 8-PSK constellation
constellation = exp(1j * 2 * pi * symbols / M);

% Passband waveform
t = 0:Ts:(symbol_count * symbol_time - Ts);
s = zeros(1, length(t));

for i = 1:symbol_count
    idx = (i-1)*fs + 1 : i*fs;
    phase = angle(constellation(i));
    s(idx) = cos(2*pi*fc*t(idx) + phase);
end

% Time Domain Signal
figure;
plot(t, s);
xlabel('Time (s)');
ylabel('Amplitude');
title('8-PSK Passband Signal');
```




```
% Constellation
figure;
plot(real(constellation), imag(constellation), 'bo');
xlabel('In-phase (I)');
ylabel('Quadrature (Q)');
title('8-PSK Constellation Diagram');
grid on;
axis([-1.5 1.5 -1.5 1.5]);
```




```
% PSD
figure;
nfft = 1024;
[psd_vals, f] = periodogram(s, [], nfft, 1/Ts);
plot(f, 10*log10(psd_vals));
xlabel('Frequency (Hz)');
ylabel('Power/Frequency (dB/Hz)');
title('PSD of 8-PSK Signal');
xlim([0, 5]);
```




```
% Bit Rate and Bandwidth
bit_rate = N / (symbol_count * symbol_time); % bits/sec
null_to_null_bw = 2 / symbol_time; % ~2 Hz for 1s symbol duration

disp(['Bit rate = ', num2str(bit_rate), ' bps']);
```

```
Bit rate = 3 bps
```

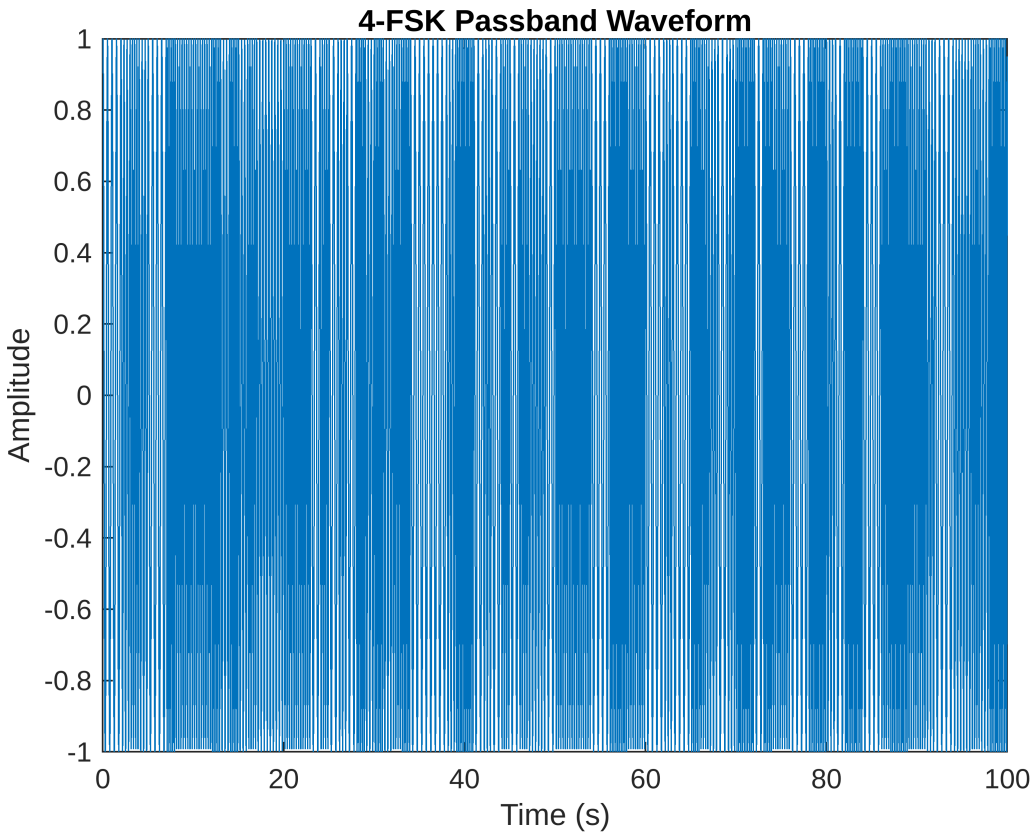
```
disp(['Null-to-null Bandwidth = ', num2str(null_to_null_bw), ' Hz']);
```

```
Null-to-null Bandwidth = 2 Hz
```

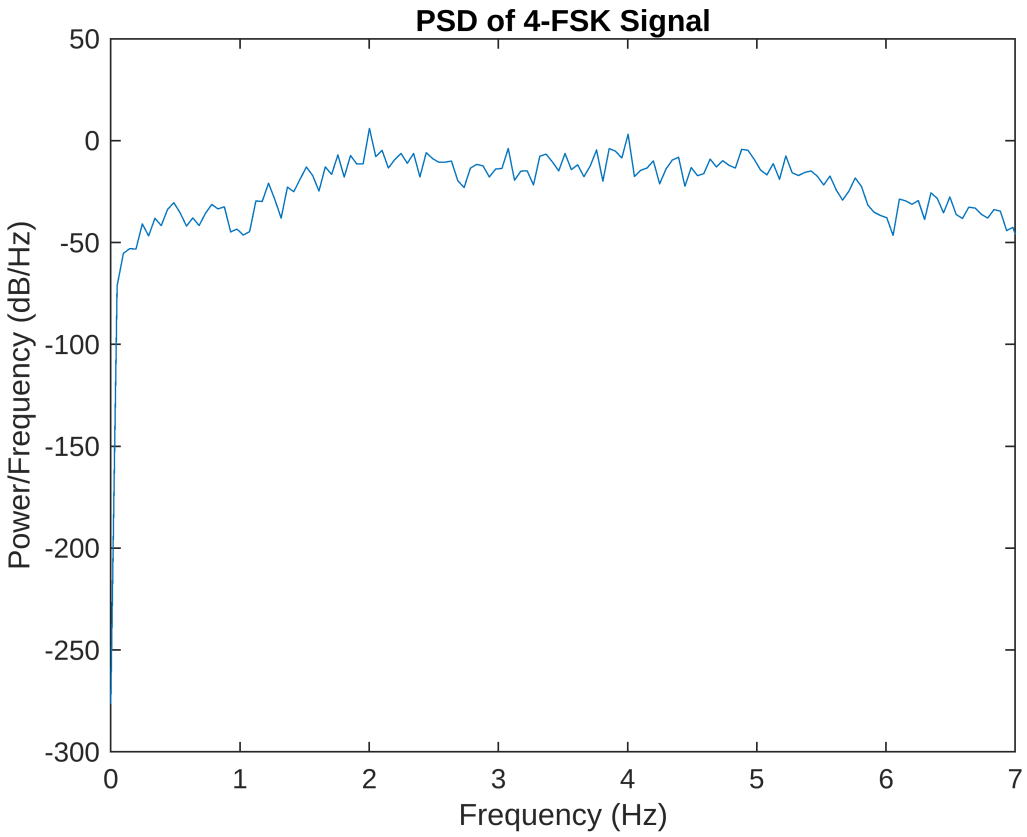
4. For a sequence of 200 bits obtain the waveform corresponding to 4-FSK. Again record the bit rate and the null to null bandwidth.

```
N_bits = 200;
bits_per_symbol = 2;
M = 2^bits_per_symbol;
N_symbols = N_bits / bits_per_symbol;
symbol_time = 1;
fs = 100; % Samples per second
samples_per_symbol = fs * symbol_time;
Ts = 1/fs;

% Bit to Symbol Mapping
```

```
bits = randi([0 1], 1, N_bits);
symbols = zeros(1, N_symbols);
for i = 1:N_symbols
    group = bits((i-1)*2 + 1 : i*2);
    symbols(i) = bin2dec(num2str(group));
end

% Frequency Mapping
f_base = 2;
delta_f = 1;
f_tones = f_base + (0:M-1) * delta_f;

% Time Vector and Signal Generation
t = 0:Ts:(N_symbols * symbol_time - Ts);
s = zeros(1, length(t)); % FSK signal

for i = 1:N_symbols
    idx = (i-1)*samples_per_symbol + 1 : i*samples_per_symbol;
    freq = f_tones(symbols(i)+1);
    s(idx) = cos(2 * pi * freq * t(idx));
end

% Time-Domain Waveform
figure;
plot(t, s);
xlabel('Time (s)');
ylabel('Amplitude');
title('4-FSK Passband Waveform');
```




```
% PSD
figure;
nfft = 2048;
[psd_vals, f] = periodogram(s, [], nfft, fs);
plot(f, 10*log10(psd_vals));
xlabel('Frequency (Hz)');
ylabel('Power/Frequency (dB/Hz)');
title('PSD of 4-FSK Signal');
xlim([0, max(f_tones)+2]);
```




```
% Bit Rate and Null-to-Null Bandwidth
bit_rate = N_bits / (N_symbols * symbol_time);
null_to_null_bw = (M-1) * delta_f;

disp(['Bit rate = ', num2str(bit_rate), ' bps']);
```

```
Bit rate = 2 bps
```

```
disp(['Null-to-null Bandwidth = ', num2str(null_to_null_bw), ' Hz']);
```

```
Null-to-null Bandwidth = 3 Hz
```

5. Compare the bit rate, null to null bandwidth, and signal constellation for the four schemes discussed above (note that it is not possible to visualize the constellation for 4-FSK).

Answer:

Bit rate (bps):

- QPSK: 2
- 16-QAM: 4
- 8-PSK: 3
- 4-FSK: 2

Null-to-Null Bandwidth (Hz):

- QPSK: 2
- 16-QAM: 4
- 8-PSK: 3
- 4-FSK: 3

Signal Constellation:

- QPSK: 4 points on unit circle (90° apart)
- 16-QAM: 16 grid points (amplitude + phase)
- 8-PSK: 8 points on circle (45° apart)
- 4-FSK: Not possible

Inference:

- Bit Rate increases with bits per symbol.
- PSK/QAM are bandwidth efficient, especially QAM.
- FSK bandwidth depends on frequency spacing **(**Δf**)** [Wider spacing → better orthogonality but wider bandwidth].
- QPSK, 16-QAM and 8-PSK have clear constellation diagrams in I-Q plane; FSK does not, as it's frequency-based.