DSP lab 02

Name: Basil Khowaja bk08432

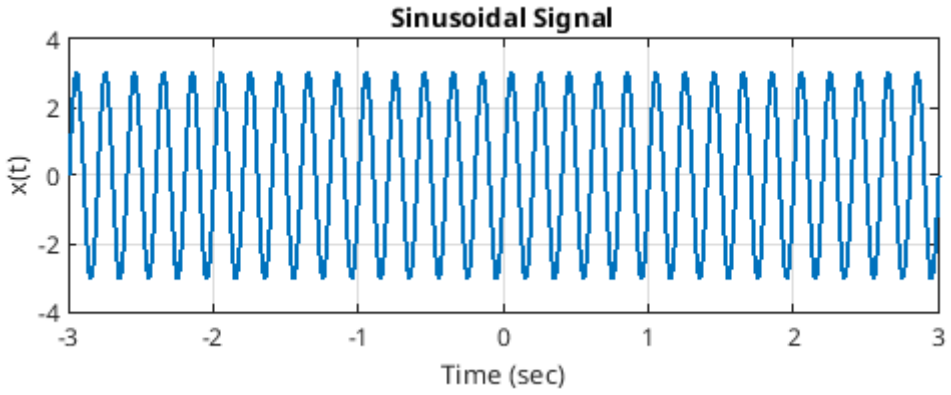
Activity 1)

Task a) Obtain the spectrum of the periodic sinusoidal signal, x1 (t) = A sin(2πf0 t).

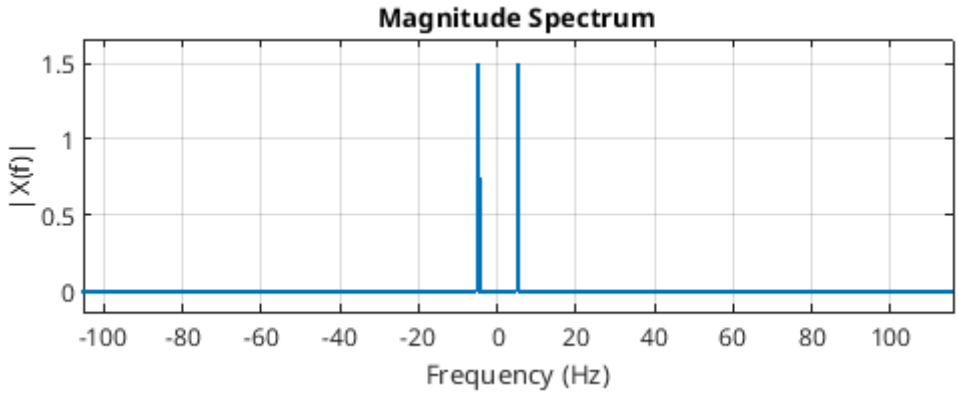
code:

```
% task a:
clc; clear; close all

A = 3;
f0 = 5;
tmax = 3;
tmin = -3;
Ts = 0.001;
fs = 1/Ts;

time = tmin:Ts:tmax;
N = length(time);

xt = A * sin(2*pi*f0*time);

Xs = fft(xt, N);
X = Ts*Xs/(tmax-tmin);
X = fftshift(X);
freq = linspace(-fs/2, fs/2, N);

figure;
subplot(211)
plot(time, xt, 'linewidth', 2);
grid on;
xlabel('Time (sec)');
ylabel('x(t)');
title('Sinusoidal Signal');

subplot(212)
plot(freq, abs(X), 'linewidth', 2);
grid on;
xlabel('Frequency (Hz)');
ylabel('|X(f)|');
title('Magnitude Spectrum');
```

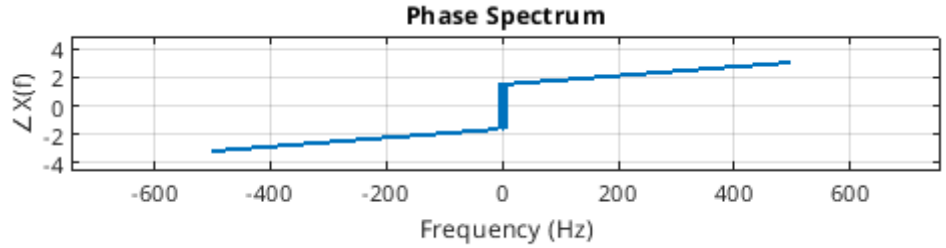
Generated sinusoidal signal:



It's corresponding magnitude spectrum:



and its corresponding phase spectrum:



observation for task a)

In this task, we created a sine wave signal using the equation x1(t)=Asin(2πf0t) and saw its frequency spectrum. The time-domain graph shows a smooth wave that moves up and down at a regular rate, just like a normal sine wave. When we applied the Fourier Transform, we got two sharp peaks in the frequency spectrum at f=±f0. This confirms that the signal contains only one frequency component. There is no extra noise or other frequencies present. Since the sine wave is

centered around zero, there is no DC component in the spectrum. Results show that that a simple sine wave in the time domain always gives two distinct spikes in the frequency domain.
Seeing the phase spectrum, one finds a linear trend with a clear discontinuity—jump—at the origin (0 Hz). The nature of the sine function causes this jump; Euler's identity allows one to translate the sine function's character into terms of complex exponentials. Two frequency components at ±f0±f0 thus reflect the observed peaks in the magnitude spectrum. With a phase shift of π/2π/2 relative to a cosine wave, the phase spectrum at these places is almost ±π2±2π. The phase spectrum's linear character implies that the signal is a time-shifted form of an ideal sine wave in which the shift coincides with the phase slope. The way the Fourier transform manages symmetry in the negative and positive frequency components produces a phase gap at zero frequency. Given that a sine wave is an odd function, its Fourier transform produces phase shifts wrapped around that produce the noted phase jump. With the expected phase shifts resulting from the representation of the sine function in the frequency domain, the phase spectrum generally helps validate the sinusoidal character of the original signal and reveals that it is centered around f0=±5f0=±5 Hz.

1. How does the frequency shift affect the position of the delta functions?

The frequency shift changes the position of the peaks in the frequency spectrum. we saw two peaks at f=±f0, which means the sine wave contains only one frequency component at f0. If we increase the frequency f0, the peaks will move more away from each other, which means that the wave oscillates faster in the time domain. If we decrease f0, the peaks will come closer to each other, which shows that the wave oscillates more slowly. If a signal is having multiple frequencies, shifting the entire signal in the time domain will move all the peaks by the same unit in the frequency domain. This is how frequency modulation works in many communication systems, where signals are shifted to different frequencies for transmission. Like each service has a separate frequency band FM radio etc.

2. What happens if the amplitude A is varied in terms of the frequency spectrum?

The amplitude A of the sine wave effects the height of the peaks in the frequency spectrum but does not change their position. If we increase A, the sine wave heights increases in the time domain, meaning the signal has more energy than previous. This increase in energy appears in the frequency domain as taller peaks at f=±f0. If we decrease A, the sine wave becomes smaller, and the peaks in the frequency spectrum also become shorter, but location of the peaks will remain the same because we havent changed the frequency.

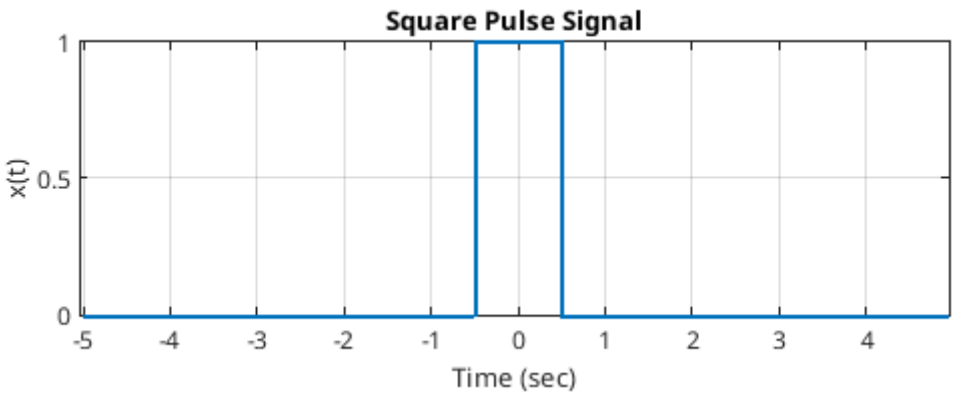
Task b)

code:

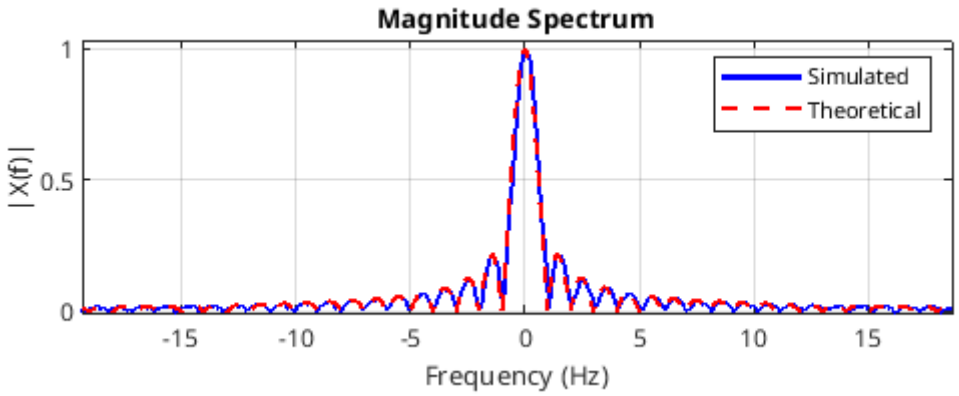
```
clc; clear; close all

A = 1;
tau = 1;
tmax = 5;
tmin = -5;
Ts = 0.01;
fs = 1/Ts;

time = tmin:Ts:tmax;
```

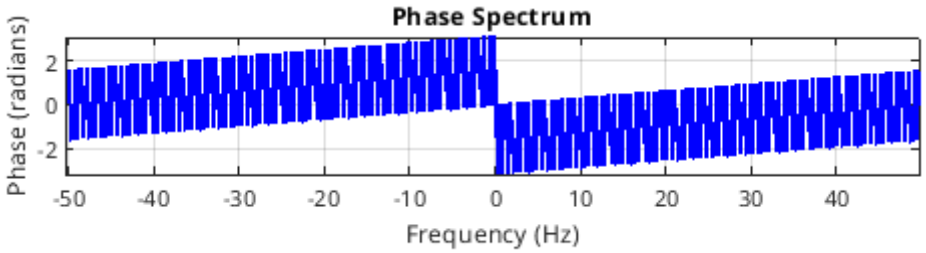
```
N = length(time);

xt = zeros(1, N);
xt(abs(time) < tau/2) = A;

Xs = fft(xt, N);
X = Ts*Xs;
X = fftshift(X);

freq = linspace(-fs/2, fs/2, N);
f_theory = freq;
X_theory = A*tau*sinc(tau*f_theory);

figure;
subplot(211)
plot(time, xt, 'linewidth', 2);
grid on;
xlabel('Time (sec)');
ylabel('x(t)');
title('Square Pulse Signal');


subplot(212)
plot(freq, abs(X), 'b', 'linewidth', 2);
hold on;
plot(f_theory, abs(X_theory), 'r--', 'linewidth', 1.5);
grid on;
xlabel('Frequency (Hz)');
ylabel('|X(f)|');
title('Magnitude Spectrum');
legend('Simulated', 'Theoretical');
```

generated non periodic square pulse signal:



it’s corresponding magnitude spectrum:





Observation of output:

We first created a square pulse signal, which stays at 1 for a short time and 0 everywhere else. When we plotted it, we saw a simple rectangular pulse in the time domain. Then, we used the Fourier Transform to find its frequency spectrum.

After applying the Fourier Transform, we got a sinc-shaped curve in the frequency domain. This means that the square pulse is made up of many different frequency components. We also plotted the theoretical sinc function, and it matched well with our simulated result. This confirmed that our calculations were correct.

We then observed that changing the pulse width affects the frequency spread. If the pulse is shorter in time, its frequency spectrum becomes wider, meaning more high-frequency components are present. If the pulse is longer in time, its frequency spectrum becomes narrower, meaning it contains fewer high frequencies.

The square pulse signal has a linear trend over the frequency axis in its phase spectrum. This behavior arises from the sinc function in the magnitude spectrum obtained by the Fourier Transform of a square pulse producing a linear phase shift in the phase spectrum. Rather than the actual power of the signal, the amplitude in this diagram shows the phase shift or delay of every frequency component. The phase spectrum displays how various frequency components are aligned in phase, unlike the magnitude spectrum, which shows the signal's frequency component presence.

One obvious aspect of the story is the abrupt zero frequency leap. A sign flip in phase causes this; this is a natural feature of the Fourier Transform for symmetric signals. The phase moves abruptly as the frequency moves from positive to negative, producing a discontinuity at f=0f=0. This is expected and points not toward a mistake.

1. What is the physical meaning of the sinc function in terms of signal bandwidth?

The sinc function shows how the pulse spreads in different frequencies. The middle part of the sinc function has the most energy, and the side parts have less energy. The width of the sinc function tells us how much of the signal is in high or low frequencies. If the sinc function is wide, the signal has more low-frequency components. If it is narrow, the signal has more high-frequency components.

2. How would you modify the pulse to alter its spectral characteristics (e.g., changing the width of the pulse)?

If we make the pulse wider, the frequency range becomes narrower, meaning the signal has fewer high-frequency components. If we make the pulse shorter, the frequency range becomes wider, meaning the signal has more high-frequency components. Another way to change the signal is by increasing or decreasing the amplitude. This makes the signal stronger or weaker, but it does not change the frequency range.

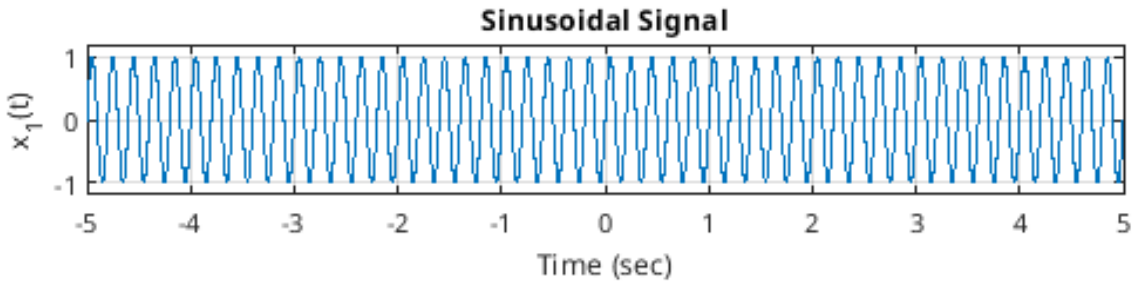
task c)

code:

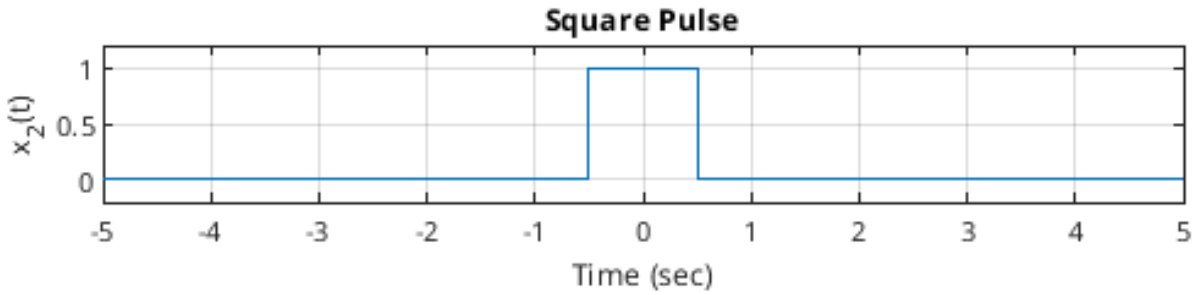
```
A = 1;
f0 = 5;
tau = 1;
tmax = 5;
tmin = -5;
Ts = 0.001;
fs = 1/Ts;
time = tmin:Ts:tmax;
N = length(time);
x1 = A * sin(2*pi*f0*time);
x2 = zeros(1, N);
x2(abs(time) < tau/2) = 1;
xt = x1 .* x2;
Xs = fft(xt, N);
X = Ts*Xs;
X = fftshift(X);
freq = linspace(-fs/2, fs/2, N);
figure;
subplot(3,1,1)
plot(time, x1, 'linewidth', 1);
grid on;
xlabel('Time (sec)');
ylabel('x_1(t)');
title('Sinusoidal Signal');
ylim([-1.2 1.2]);

subplot(3,1,2)
```

```
plot(time, x2, 'linewidth', 1);
grid on;
xlabel('Time (sec)');
ylabel('x_2(t)');
title('Square Pulse');
ylim([-0.2 1.2]);

subplot(3,1,3)
plot(time, xt, 'linewidth', 2);
grid on;
xlabel('Time (sec)');
ylabel('x_3(t)');
title('RF Pulse Signal');
ylim([-1.2 1.2]);

figure;
plot(freq, abs(X), 'linewidth', 2);
grid on;
xlabel('Frequency (Hz)');
ylabel('|X(f)|');
title('Spectrum of RF Pulse');
```

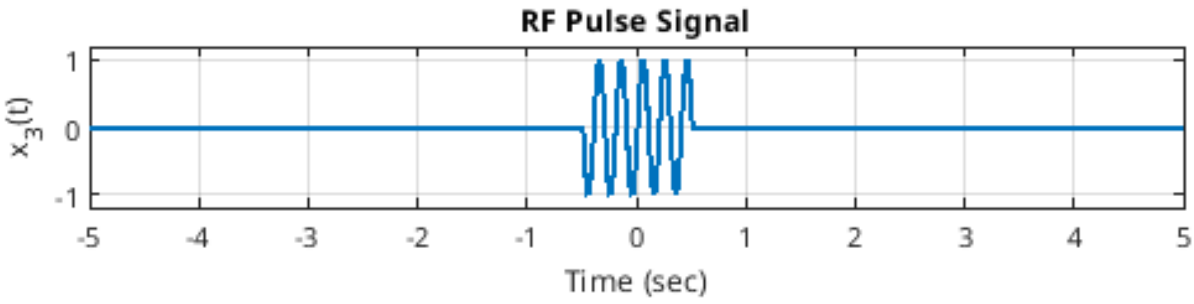
Generated sinusoidal signal x1(t):

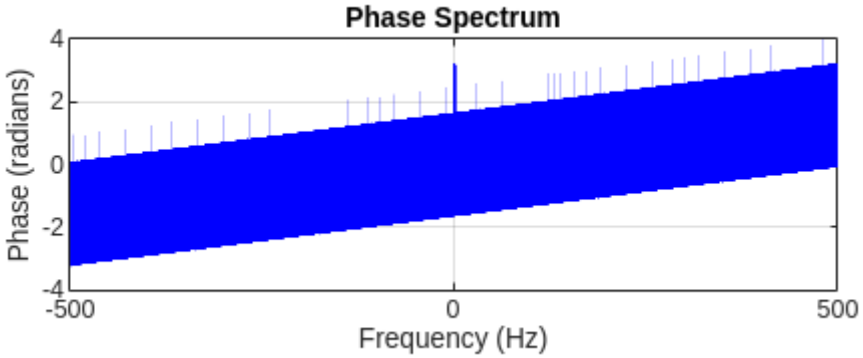


generated Square pulse signal x2(t):

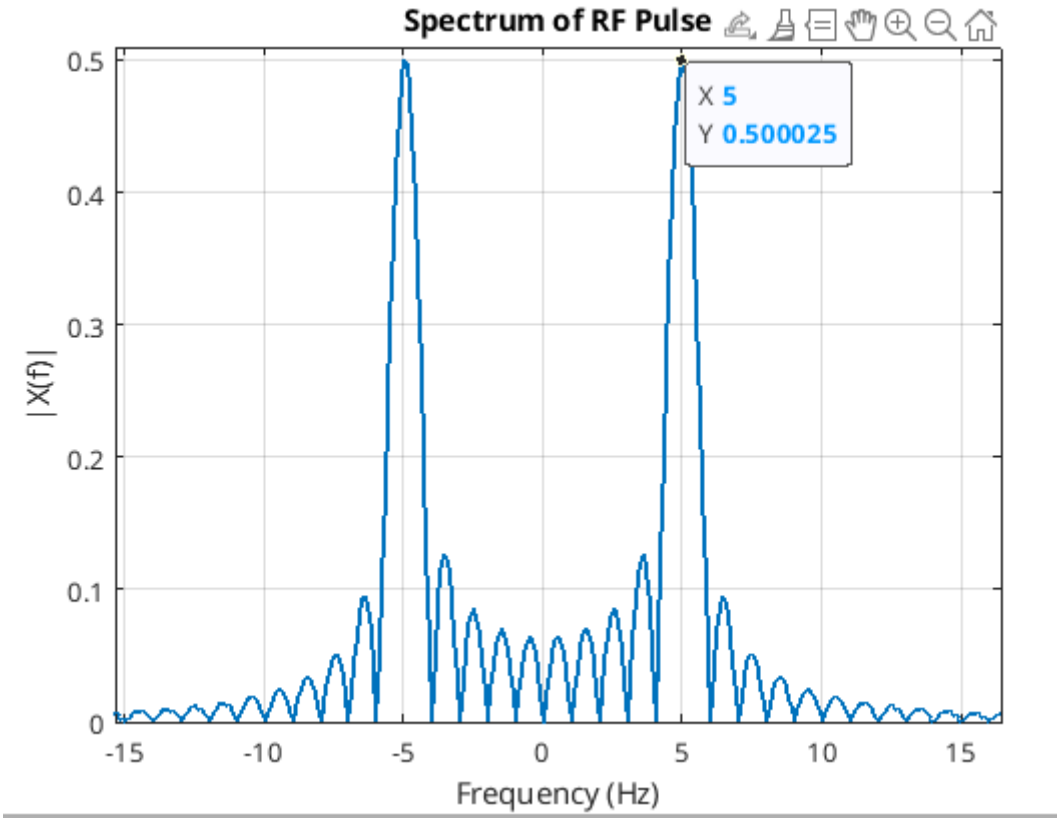


Resultant Rf pulse signal x3(t):





Output magnitude spectrum of this RF pulse signal x3(t):



Observation:

First, we created a sinusoidal signal x1(t) using a sine function with frequency f0=5 Hz. When we plotted it, we saw a smooth wave oscillating between positive and negative values. then we created a square pulse signal x2(t), which was 1 for a short time and 0 everywhere else. When we plotted it, we saw a rectangular pulse that was centered around zero. Then, we multiplied the sinusoidal signal x1(t) with the square pulse x2(t) to get the RF pulse signal x3(t). The result was that the sinusoidal wave was present only where the pulse was 1, and everywhere else it was zero. When we plotted it, we saw a sine wave appearing only inside the pulse duration. Finally, we applied the Fourier Transform to x3(t) to see its frequency spectrum. The spectrum showed two main peaks at f=±5 Hz, matching the frequency of the sine wave. It also had a sinc-like shape, which comes from the multiplication with the pulse. This means that the spectrum was formed by combining the frequency properties of both the sine wave and the pulse. The results matched the expected theoretical results.

1. How does the frequency f0 of the sinusoidal signal interact with the width of the pulse τ?

The frequency f0 of the sinusoidal signal will choose where the peaks appear in the frequency spectrum while the width of the pulse τ controls how much spreaded is the spectrum If the pulse is wide then the spectrum becomes narrow which means that the signal is focused around ±f0. If the pulse is short then the spectrum becomes wide, which means that it contains more frequency components. This happens because a short pulse has fast changes which will then require higher frequencies to represent it. So, the width of the pulse decides how much the frequencies are spread out and the sine wave frequency f0 will help in finding the central position of the spectrum.

2. What would happen to the spectrum if the amplitude A is increased?

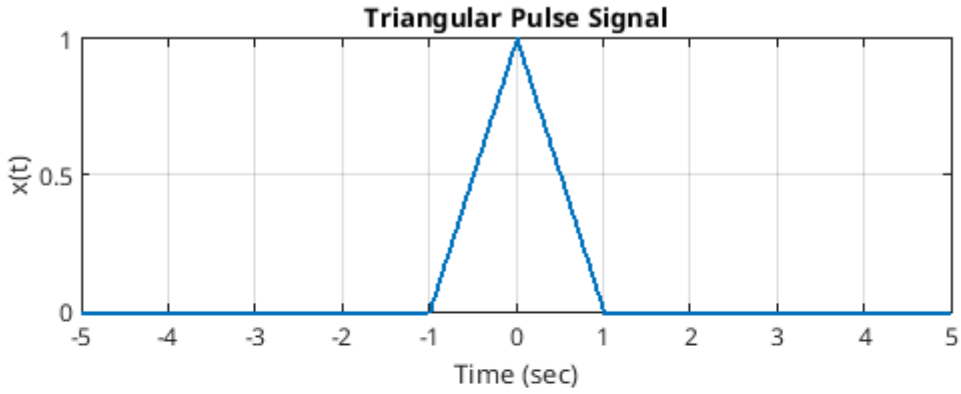
If we make a signal bigger, the frequency peaks in its spectrum will get taller, but won't move to different spots. This happens because a bigger signal is stronger, which means more energy at each frequency. Making the signal smaller does the opposite which is that the peaks get shorter since there's less energy. But no matter if we make the signal bigger or smaller, the frequencies don't change at all. Basically, changing how big the signal is only affects the height of the peaks, not where they show up.

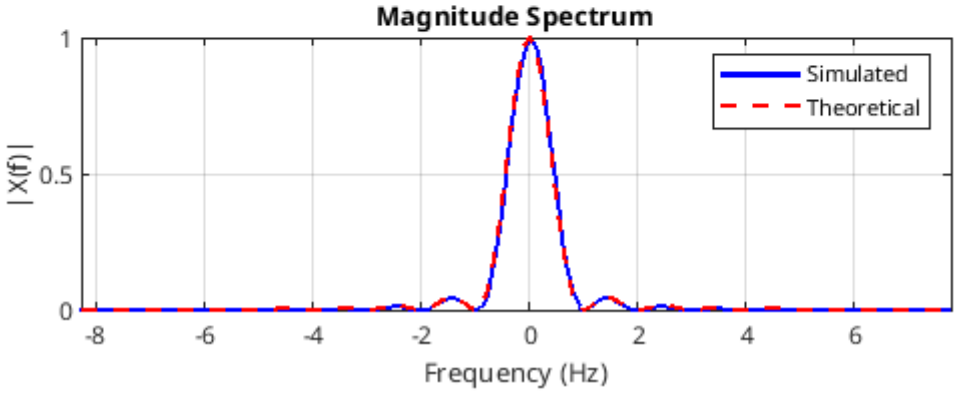
Task d)

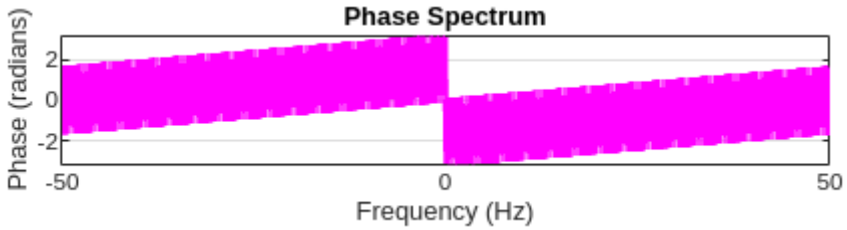
code:

```
%task d:

clc; clear; close all

A = 1;
tau = 1;
tmax = 5;
tmin = -5;
Ts = 0.01;
fs = 1/Ts;
```

```
time = tmin:Ts:tmax;
N = length(time);

xt = zeros(1, N);
for i = 1:N
    t = time(i);
    if -tau <= t && t < 0
        xt(i) = A/tau * (tau + t);
    elseif 0 <= t && t < tau
        xt(i) = A/tau * (tau - t);
    end
end

Xs = fft(xt, N);
X = Ts*Xs;
X = fftshift(X);

freq = linspace(-fs/2, fs/2, N);

f_theory = freq;
X_theory = A*tau*(sinc(tau*f_theory)).^2;

figure;
subplot(211)
plot(time, xt, 'linewidth', 2);
grid on;
xlabel('Time (sec)');
ylabel('x(t)');
title('Triangular Pulse Signal');


subplot(212)
plot(freq, abs(X), 'b', 'linewidth', 2);
hold on;
plot(f_theory, abs(X_theory), 'r--', 'linewidth', 1.5);
grid on;
xlabel('Frequency (Hz)');
ylabel('|X(f)|');
title('Magnitude Spectrum');
legend('Simulated', 'Theoretical');
```







Observation:

First, I created a triangular pulse signal that starts at zero, rises to a peak, and then comes back down to zero. When I plotted it, I saw a smooth triangular shape, unlike a square pulse that has sharp edges. Then, I applied the Fourier Transform to find its frequency spectrum. The result showed a squared sinc function (sinc2\text{sinc}^2), meaning the signal's energy is more concentrated in a smaller range of frequencies. I compared this with the theoretical spectrum, and both matched closely, confirming that the result was correct. From this, I understood that a triangular pulse has a narrower frequency range than a square pulse because it changes more smoothly over time. Since there are no sudden jumps, it has fewer high-frequency components. This means that a triangular pulse is more bandwidth-efficient, making it useful in situations where I need to limit high-frequency noise or interference.

1. How does the width of the triangular pulse τ affect its frequency spectrum?

First, I changed the width of the triangular pulse τ and saw how it affected the frequency spectrum. When I made the pulse wider, the frequency spectrum became narrower, meaning the signal had

fewer high-frequency components. When I made the pulse narrower, the frequency spectrum became wider, meaning more high-frequency components were present. This happens because a wider pulse changes more smoothly over time, so it does not need many high frequencies. A narrower pulse has sharper changes, which means it needs more high frequencies to represent it.

2. What is the relationship between the shape of the pulse and its frequency content?

I noticed that the shape of the pulse decides how its energy is spread in frequency. A smooth pulse in time has a narrower frequency spectrum, meaning it contains fewer high frequencies. A pulse with sharp edges or sudden changes spreads out more in frequency, meaning more high frequencies are needed. For example, the triangular pulse has a sinc-squared frequency response, which is more focused compared to the sinc function of a square pulse. This means that smooth signals in time are more efficient in frequency, while signals with sharp changes need a wider frequency range.

Activity 2)

Part a Polar-RZ)

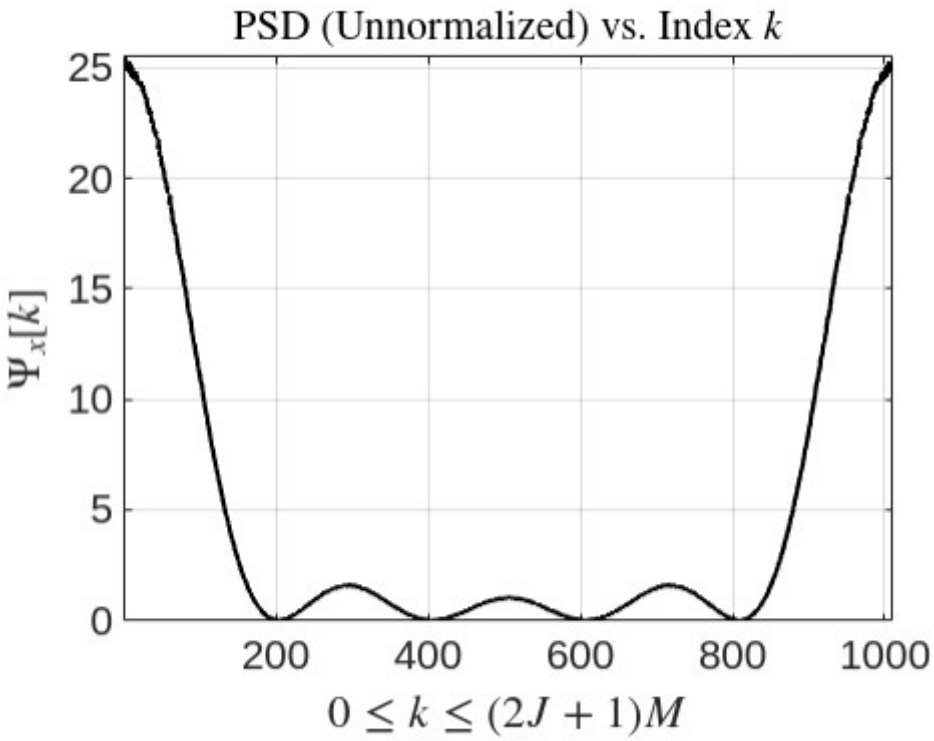
code:

```
%part a :
clc; clear; close all;
A = 1; J = 50; M = 10; L = 20000;
PSD = 0; T = 1; Ts = T/M; fs = 1/Ts;

for jj = 1:L
   sNt = [];
   Bits = (sign(randn(1,2*J+1)) + 1) / 2;

   for ii = 1:2*J+1
      if Bits(ii) == 1
         sNt = [A*ones(1,M/2), zeros(1,M/2), sNt];  % RZ: Half-duration active
      else
         sNt = [-A*ones(1,M/2), zeros(1,M/2), sNt];
      end
   end
   PSD = PSD + abs(fft(sNt)).^2;
end

PSD = PSD / L / (2*J+1) / T;
LPSD = length(PSD);

% Plot 1: Unnormalized PSD vs. Index k
figure(1);
plot(PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$0\leq k\leq (2J+1)M$', 'FontSize', 16, 'Interpreter', 'latex');
```

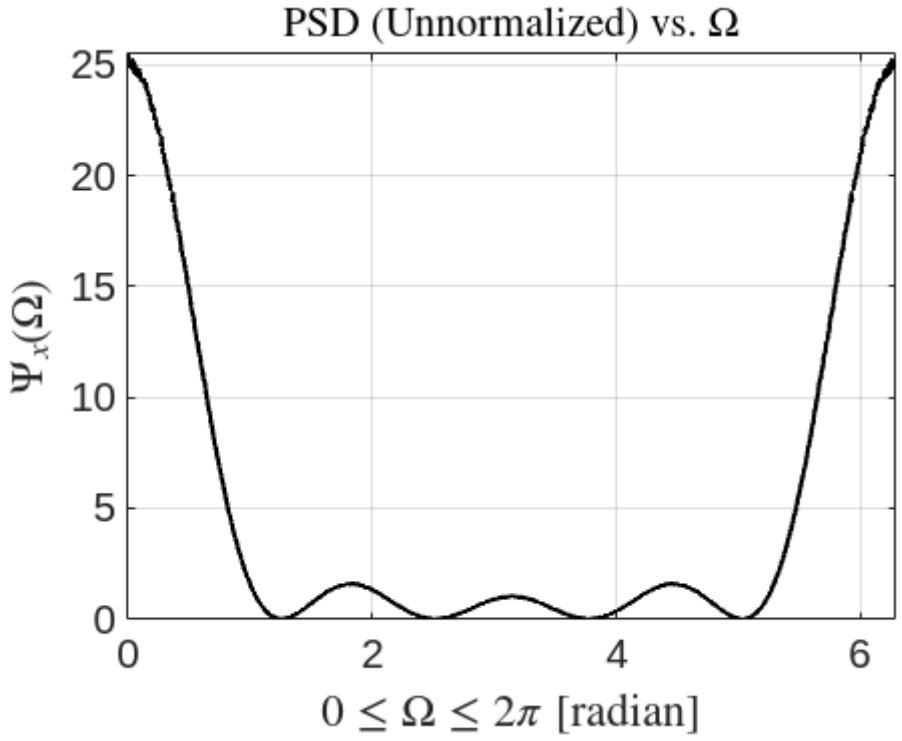
```
title('PSD (Unnormalized) vs. Index $k$', 'FontSize', 14, 'Interpreter', 'latex');
ylabel('$\Psi_x[k]$', 'FontSize', 16, 'Interpreter', 'latex');

% Plot 2: PSD vs. Normalized Frequency (DTFT-based)
DTFT_xaxis = linspace(0, 2*pi, LPSD);
figure(2);
plot(DTFT_xaxis, PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$0 \leq \Omega \leq 2\pi$ [radian]', 'FontSize', 16, 'Interpreter', 'latex');
title('PSD (Unnormalized) vs. $\Omega$', 'FontSize', 14, 'Interpreter', 'latex');
ylabel('$\Psi_x(\Omega)$', 'FontSize', 16, 'Interpreter', 'latex');

% Plot 3: Normalized and Two-Sided PSD (CTFT-based)
fT = linspace(-T*fs/2, T*fs/2, LPSD);  % Normalized frequency axis
Area = sum(PSD) * (fT(2) - fT(1));
CTFT_PSD = PSD / Area;
CTFT_PSD = [fliplr(CTFT_PSD(end:-1:floor(LPSD/2))), CTFT_PSD(1:floor(LPSD/2)-1)];

figure(3);
plot(fT, CTFT_PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$-Tf_s/2 \leq Tf \leq Tf_s/2$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$\Psi_x(f)$', 'FontSize', 16, 'Interpreter', 'latex');
xlim([-fs/2 fs/2]);
title('PSD vs. $Tf$ [Normalized and Two-Sided]', 'FontSize', 14, 'Interpreter', 'latex');

% Plot 4: Normalized Semilog Plot
figure(4);
semilogy(fT, CTFT_PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14);
xlabel('$-Tf_s/2 \leq Tf \leq Tf_s/2$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$\Psi_x(f)$ [dB]', 'FontSize', 16, 'Interpreter', 'latex');
ylim([1e-3 1]);
xlim(T*[-fs/2 fs/2]);
title('Log PSD vs. $Tf$ [Normalized and Two-Sided]', 'FontSize', 14, 'Interpreter', 'latex');
grid on; axis tight;
```

Results obtained:

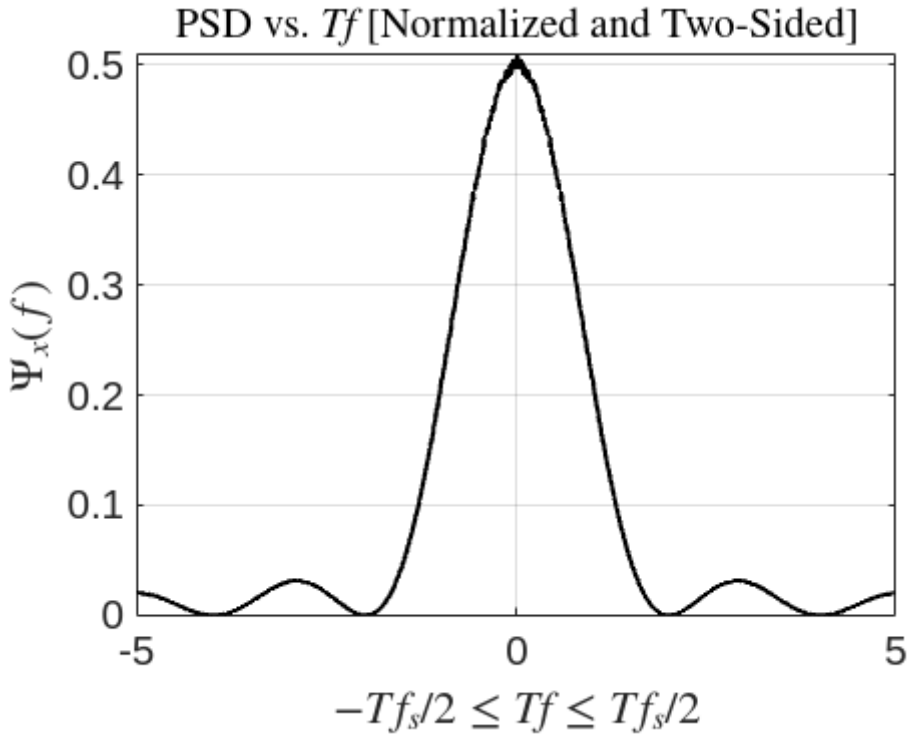
plot 1:



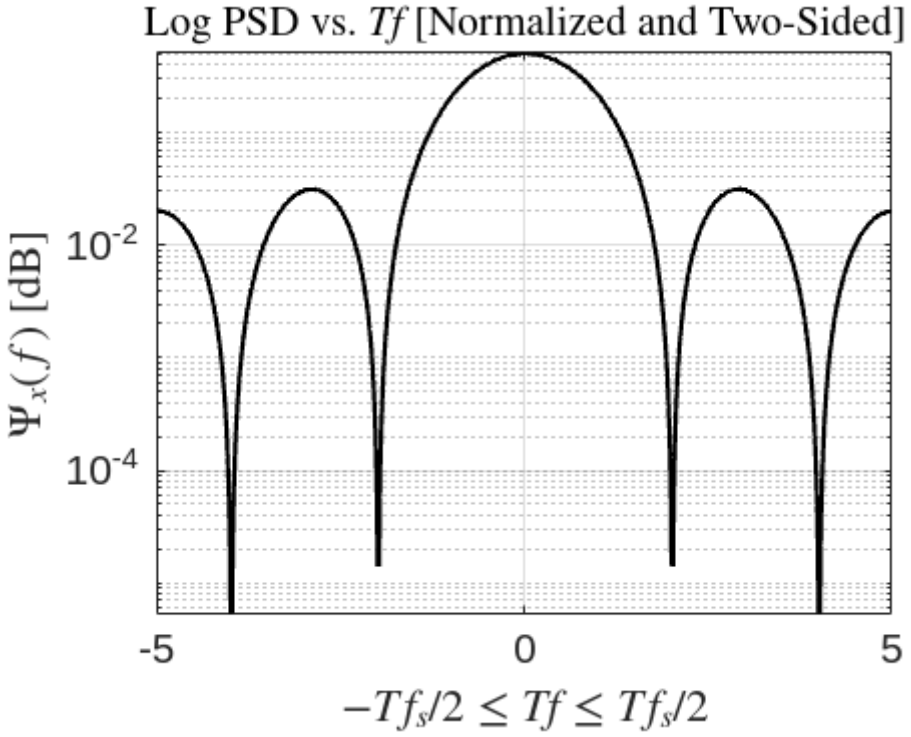
plot b obtained:



plot c obtained:



plot D obtained:



observations:

This work examined Fourier Transform-based Power Spectral Density (PSD) of a return-to-zero (RZ) signal. The four charts show several approaches of viewing the PSD and the frequency distribution of the energy. **The original Polar NRZ code assigned a constant value of +A or −A for a full bit duration. To modify it for Polar RZ, I changed the waveform so that a bit is only active (nonzero) for half of the bit duration and zero for the remaining half. This was done by updating the pulse generation logic inside the loop, where a '1' was represented as +A for half the duration and zero for the rest, while a '0' was represented as −A for half the duration and zero for the rest.** This change introduced more frequent transitions in the signal

First plot: unnormalized PSD against index k
Corresponding with the discrete frequency components, this plot displays the raw, unnormalized PSD as a function of index k. The sinc-like form of the curve indicates that the power usually concentrates in the center frequencies and falls toward the edges. This suggests that most of the energy in the RZ signal is focused in the lower frequency range and so has a narrow bandwidth.

Second plot:
Understanding the behavior of the spectrum in a continuous domain requires knowledge of the PSD expressed in terms of normalized frequency ΩΩ (in radians). The form stays the same as Plot 1, but this presentation facilitates comparison with theoretical projections. Whereas the side lobes show the lower-energy frequency contributions, the main lobe of the sinc function reflects the major frequency components of the signal.

Third plot:
Here the PSD is normalized and displayed in a two-sided frequency spectrum, therefore include both positive and negative frequencies. Though its form stays the same, in a continuous frequency domain interpretation of it is simpler since it is now centered at zero frequency. This indicates that the energy of the RZ signal is symmetrically distributed around zero, so verifying that it lacks DC component which means that there is no significant peak at zero frequency which is expected from an alternating signal.

Fourth plot:
This map shows the PSD using a logarithmic scale, which helps in the observation of the smaller frequency components not obviously visible in earlier charts. Here the sinc-shaped decay is more observable which indicates that while higher frequencies have much less power, much of the power is concentrated in the lower frequencies. This proves that most of the RZ signal's energy is concentrated in a given frequency range and that it is band-limited, hence lacking limitless bandwidth.

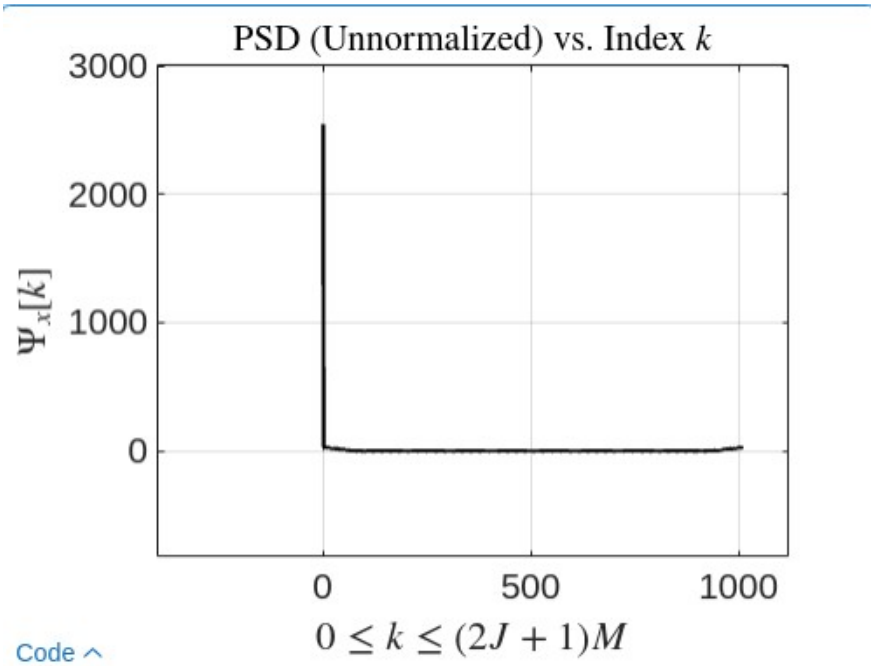
Task b Unipolar-NRZ)

code:

```
%task b:
clc; clear; close all;
A = 1; J = 50; M = 10; L = 20000;
PSD = 0; T = 1; Ts = T/M; fs = 1/Ts;

for jj = 1:L
   sNt = [];
   Bits = (sign(randn(1,2*J+1)) + 1) / 2;

   for ii = 1:2*J+1
      if Bits(ii) == 1
         sNt = [A*ones(1,M), sNt];
      else
         sNt = [zeros(1,M), sNt];
      end
   end
   PSD = PSD + abs(fft(sNt)).^2;
end


PSD = abs(PSD) / L / (2*J+1) / T;
LPSD = length(PSD);

% Plot 1: Unnormalized PSD vs. Index k
figure(1);
plot(PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$0\leq k\leq (2J+1)M$', 'FontSize', 16, 'Interpreter', 'latex');
title('PSD (Unnormalized) vs. Index $k$', 'FontSize', 14, 'Interpreter', 'latex');
ylabel('$\Psi_x[k]$', 'FontSize', 16, 'Interpreter', 'latex');

% Plot 2: PSD vs. Normalized Frequency (DTFT-based)
DTFT_xaxis = linspace(0, 2*pi, LPSD);
figure(2);
plot(DTFT_xaxis, PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$0 \leq \Omega \leq 2\pi$ [radian]', 'FontSize', 16, 'Interpreter', 'latex');
title('PSD (Unnormalized) vs. $\Omega$', 'FontSize', 14, 'Interpreter', 'latex');
ylabel('$\Psi_x(\Omega)$', 'FontSize', 16, 'Interpreter', 'latex');

% Plot 3: Normalized and Two-Sided PSD (CTFT-based)
fT = linspace(-T*fs/2, T*fs/2, LPSD);  % Normalized frequency axis
Area = sum(PSD) * (fT(2) - fT(1));
CTFT_PSD = PSD / Area;
CTFT_PSD = [fliplr(CTFT_PSD(end:-1:floor(LPSD/2))), CTFT_PSD(1:floor(LPSD/2)-1)];
```

```
figure(3);
plot(fT, CTFT_PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$-Tf_s/2 \leq Tf \leq Tf_s/2$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$\Psi_x(f)$', 'FontSize', 16, 'Interpreter', 'latex');
xlim([-fs/2 fs/2]);
title('PSD vs. $Tf$ [Normalized and Two-Sided]', 'FontSize', 14, 'Interpreter', 'latex');

% Plot 4: Normalized Semilog Plot
figure(4);
semilogy(fT, CTFT_PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14);
xlabel('$-Tf_s/2 \leq Tf \leq Tf_s/2$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$\Psi_x(f)$ [dB]', 'FontSize', 16, 'Interpreter', 'latex');
ylim([1e-3 1]);
xlim(T*[-fs/2 fs/2]);
title('Log PSD vs. $Tf$ [Normalized and Two-Sided]', 'FontSize', 14, 'Interpreter', 'latex');
grid on; axis tight;
```

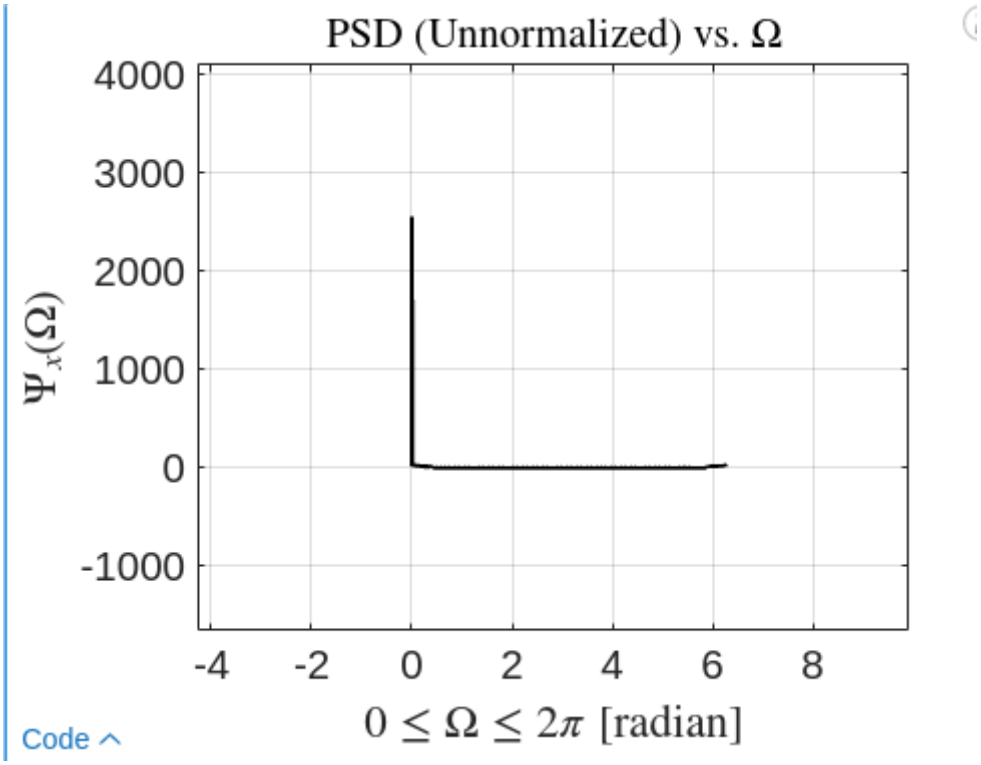
plot 1 obtained)



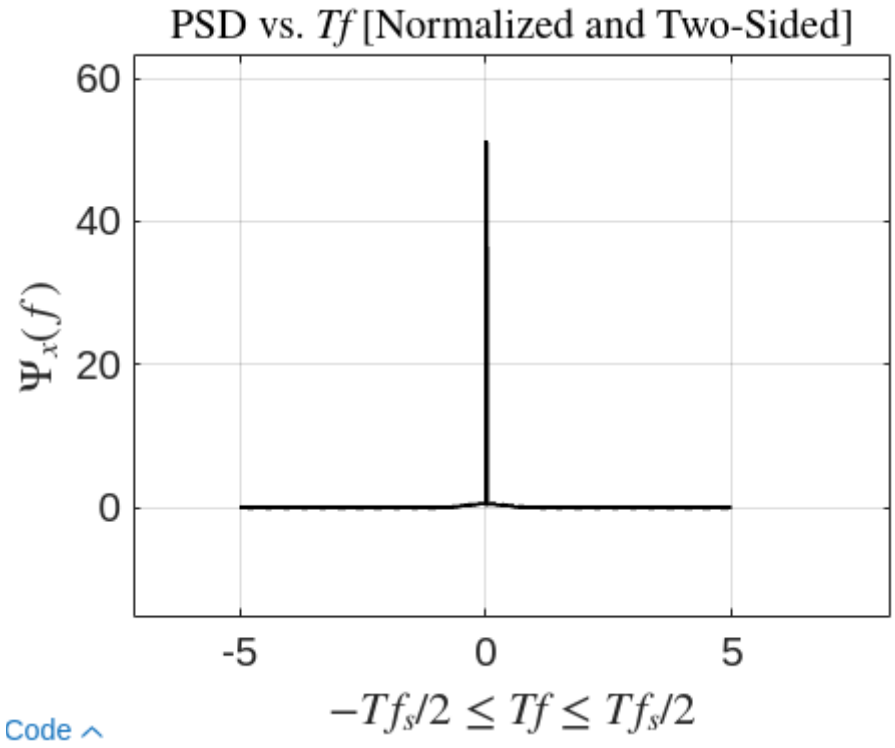
Code

```
xlim([-395 1118])
ylim([-819 3004])
```

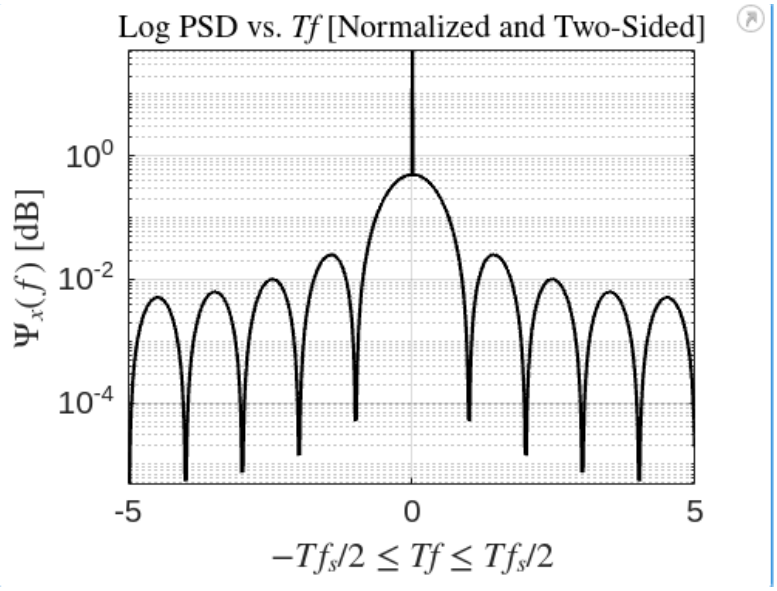
plot 2 obtained)



plot 3 obtained)



plot 4 obtained)



Observation:

For this task, I revised the Polar RZ code to apply Unipolar NRZ. Whereas a "0" is represented as zero for the whole bit duration, a binary "1" is represented in this version as +A for the whole bit length. Unipolar NRZ is distinguished from Polar NRZ, which alternates bits between +A and −A, by the presence of just positive values and zero, therefore producing a DC bias in the signal. The nonzero average value in the signal that results from DC bias defines it and causes a notable DC component (zero-frequency energy) to be present. The four graphs shown assist in better understanding the distribution of the Power Spectral Density (PSD).

Plot 1 (Unnormalized PSD vs. Index kk) shows a substantial DC component that is, a considerable fraction of the power of the signal is concentrated at zero frequency by showing the great peak at k=0. Unlike Polar NRZ, whereby the positive and negative numbers cancel one other out, Unipolar NRZ does not balance around zero. Though most of the energy stays in the low-frequency range, the remaining power radiates throughout several frequencies.

Plot 2 (PSD vs. Normalized Frequency ΩΩ) shows much higher dominance of the DC component with a strong peak at Ω=0. The sinc-like decrease in power reveals low-frequency dominance of the Unipolar NRZ signal and absence of notable high-frequency components. This happens when Unipolar NRZ has less fast transitions than Polar NRZ or RZ, so lowering its high-frequency contents.

Plot 3 (Normalized and Two-Sided PSD against TfTf) shows symmetrically both positive and negative frequencies. The great power concentration at zero frequency shows even more that Unipolar NRZ has DC bias, so it carries a constant energy level that does not add to significant information. Unipolar NRZ has a strong low-frequency presence and relatively little power at higher frequencies unlike Polar NRZ, which has more equally distributed power.
The logarithmic scale in Plot 4 (Log PSD against TfTf - Semilog Plot) emphasizes power distribution. While the sinc-shaped decline indicates most of the energy is held in the lower frequencies, the big peak at zero frequency confirms the dominating DC component. Unipolar NRZ is not bandwidth-efficient, as the fast decline in power at higher frequencies indicates does not properly use high-frequency areas.

Polar RZ to Unipolar NRZ changes the signal's frequency spread and strengthens its DC component. Clearly shown in the PSD graphs, DC bias results from the signal only taking positive values and zero not balancing around zero. **Unipolar NRZ is so ineffective for transmission since many communication systems filter out the DC component, so generating possible signal loss. Furthermore less suitable for high-speed data transmission is Unipolar NRZ since its low-frequency dominance results in ineffective utilization of bandwidth.** The graphs verify that although Unipolar NRZ is easy to use, it has major disadvantages in terms of spectral usage and power economy relative to Polar or Bipolar encoding techniques.

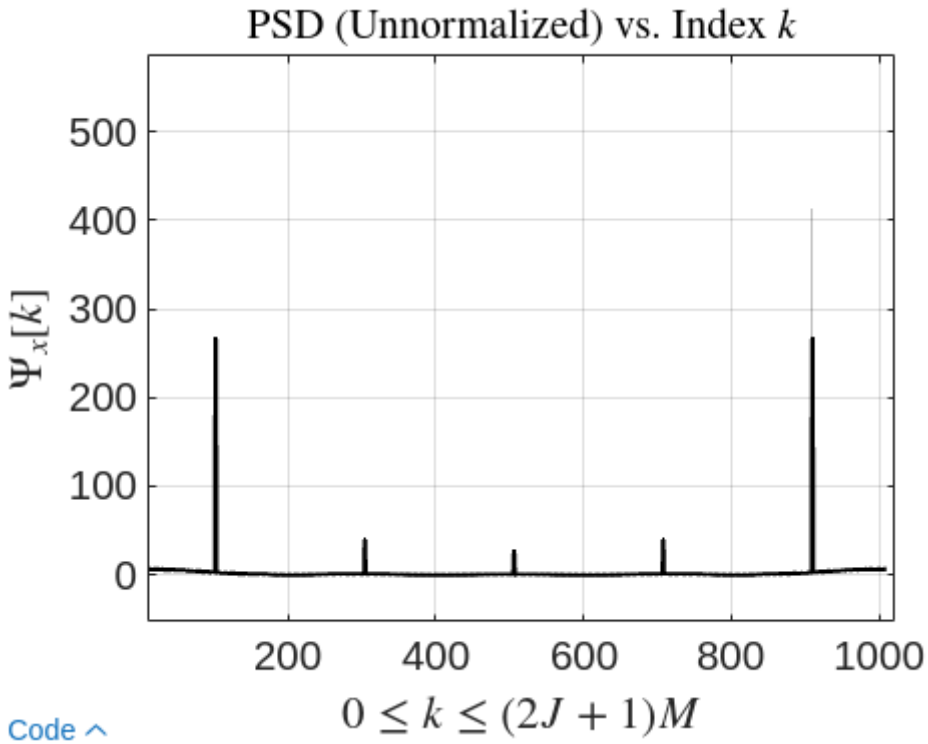
task c unipolar RZ)

code:

```
%task c:
clc; clear; close all;
A = 1; J = 50; M = 10; L = 20000;
PSD = 0; T = 1; Ts = T/M; fs = 1/Ts;

for jj = 1:L
   sNt = [];
   Bits = (sign(randn(1,2*J+1)) + 1) / 2;

   for ii = 1:2*J+1
      if Bits(ii) == 1
         sNt = [A*ones(1,M/2), zeros(1,M/2), sNt];
      else
         sNt = [zeros(1,M), sNt];
      end
   end
   PSD = PSD + abs(fft(sNt)).^2;
end

PSD = abs(PSD) / L / (2*J+1) / T;
LPSD = length(PSD);
```

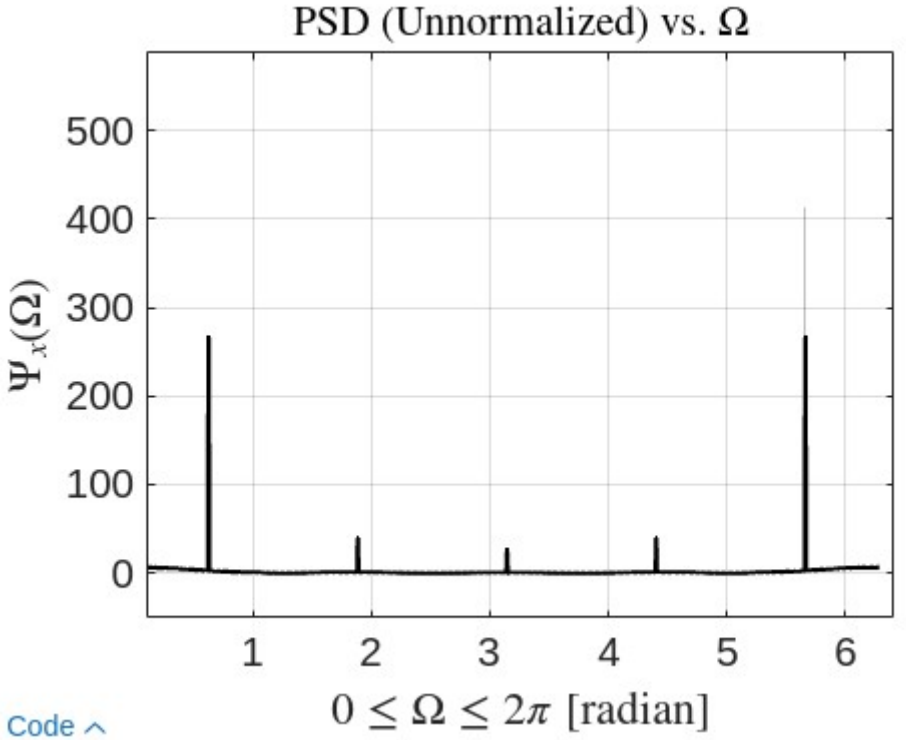
```
% Plot 1: Unnormalized PSD vs. Index k
figure(1);
plot(PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$0\leq k\leq (2J+1)M$', 'FontSize', 16, 'Interpreter', 'latex');
title('PSD (Unnormalized) vs. Index $k$', 'FontSize', 14, 'Interpreter', 'latex');
ylabel('$\Psi_x[k]$', 'FontSize', 16, 'Interpreter', 'latex');

% Plot 2: PSD vs. Normalized Frequency (DTFT-based)
DTFT_xaxis = linspace(0, 2*pi, LPSD);
figure(2);
plot(DTFT_xaxis, PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$0 \leq \Omega \leq 2\pi$ [radian]', 'FontSize', 16, 'Interpreter', 'latex');
title('PSD (Unnormalized) vs. $\Omega$', 'FontSize', 14, 'Interpreter', 'latex');
ylabel('$\Psi_x(\Omega)$', 'FontSize', 16, 'Interpreter', 'latex');

% Plot 3: Normalized and Two-Sided PSD (CTFT-based)
fT = linspace(-T*fs/2, T*fs/2, LPSD);  % Normalized frequency axis
Area = sum(PSD) * (fT(2) - fT(1));
CTFT_PSD = PSD / Area;
CTFT_PSD = [fliplr(CTFT_PSD(end:-1:floor(LPSD/2))), CTFT_PSD(1:floor(LPSD/2)-1)];

figure(3);
plot(fT, CTFT_PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$-Tf_s/2 \leq Tf \leq Tf_s/2$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$\Psi_x(f)$', 'FontSize', 16, 'Interpreter', 'latex');
xlim([-fs/2 fs/2]);
title('PSD vs. $Tf$ [Normalized and Two-Sided]', 'FontSize', 14, 'Interpreter', 'latex');

% Plot 4: Normalized Semilog Plot
figure(4);
semilogy(fT, CTFT_PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14);
xlabel('$-Tf_s/2 \leq Tf \leq Tf_s/2$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$\Psi_x(f)$ [dB]', 'FontSize', 16, 'Interpreter', 'latex');
ylim([1e-3 1]);
xlim(T*[-fs/2 fs/2]);
title('Log PSD vs. $Tf$ [Normalized and Two-Sided]', 'FontSize', 14, 'Interpreter', 'latex');
grid on; axis tight;
```

plot 1 obtained:



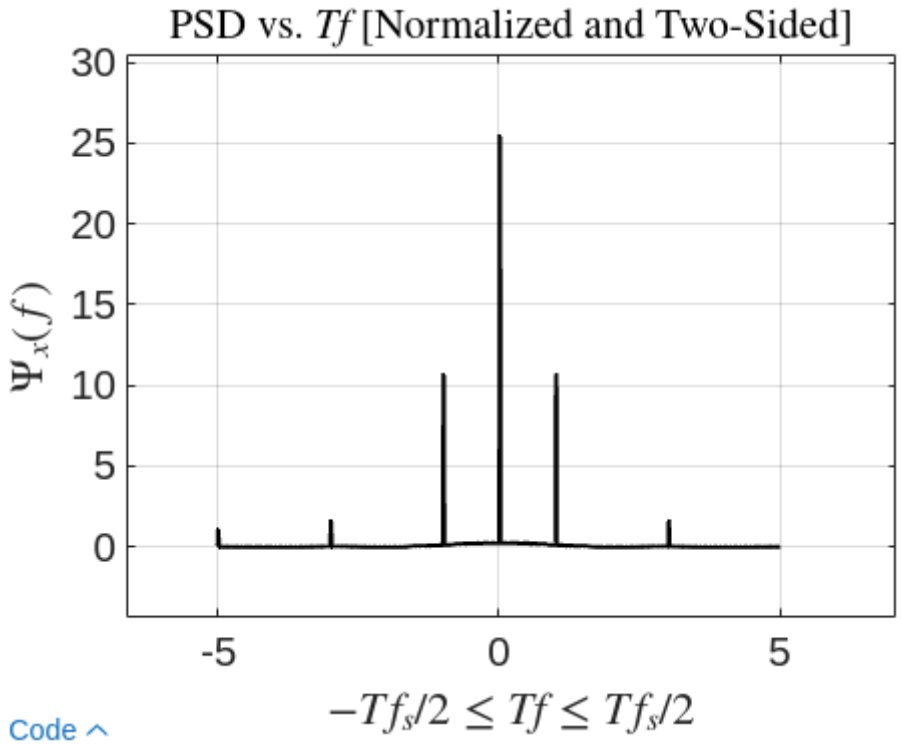
Code ^

Plot 2 obtained:



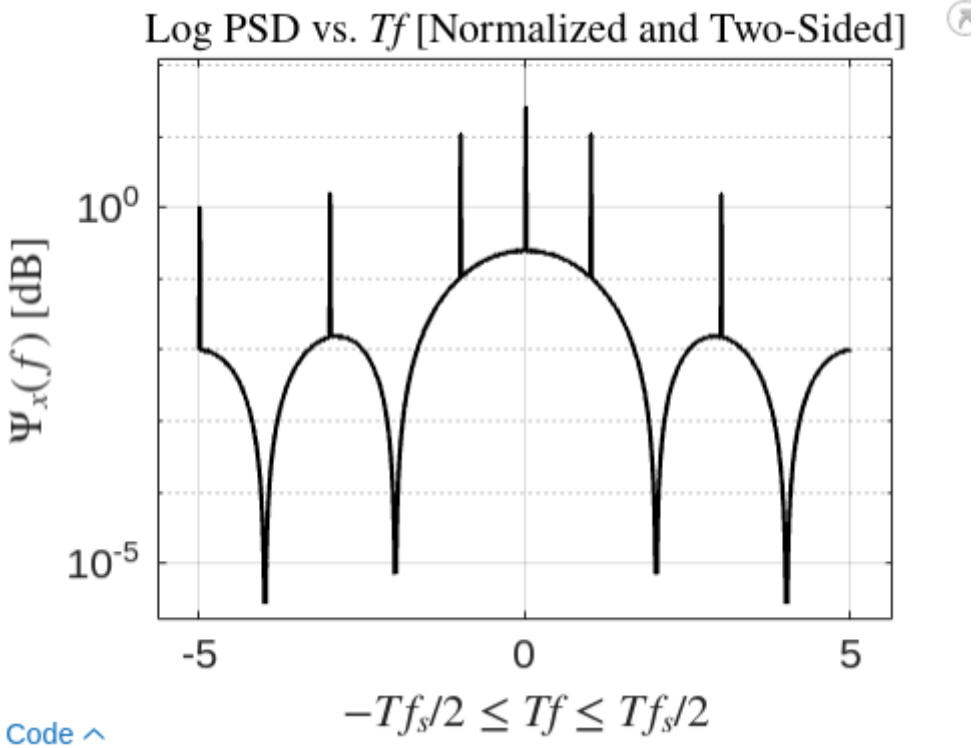
Code ^

plot 3 obtained)



Code ^

Plot 4 obtained)



Code ^

Observation:

In this task, I changed the Polar RZ code to implement Unipolar RZ, whereby a binary "0" is always represented as zero and a binary "1" is represented as +A for half of the bit duration and returns to zero for the remaining half. Unlike Unipolar NRZ, in which the signal remains at +A for the whole bit length. Unipolar RZ differs primarily from other RZ in that it adds more frequent transitions, hence raising the signal's high-frequency components. Plotting of the Power Spectral Density (PSD) offers understanding of frequency-based power distribution.

First plot: Unnormalized PSD against Index k

This graph illustrates the distribution of power among several frequency components. Unipolar RZ has a large DC component unlike Unipolar NRZ, where its persistent positive value for "1" causes a strong DC peak to develop since the signal returns to zero. Rather, repeating intervals of peaks show the fundamental frequency and its harmonic counterparts. Unipolar RZ is a periodic waveform with regular transitions, so its frequency content comprises of the base frequency and multiples of it. These peaks result from this. The first main peak results from the bit rate's corresponding fundamental frequency. The smaller peaks at higher indices indicate harmonics, therefore the signal comprises energy at several frequencies rather than only the fundamental one.

Plot 2: Normalised Frequency ΩΩ against PSD

This graph shows the PSD in continuous frequency range. The observed peaks at particular ΩΩ suggest that the signal has regular interval dominating frequency components. These peaks show the harmonics and basic frequency. Unipolar RZ has a square-like waveform, so the reason these peaks show is that the Fourier Transform of a square pulse comprises many harmonics. Unipolar RZ spreads its energy over several frequency bands due to its return-to- zero characteristic, unlike Unipolar NRZ, in which the energy was essentially concentrated at low frequencies. The periodic peaks verify that the signal consists of a basic frequency together with their higher harmonics.

Plot 3: TfTf against a normalized, two-sided PSD

This graph provides the PSD both in positive and negative frequencies and normalizes it. DC energy is clearly lowered in the tall peak at f=0 compared to Unipolar NRZ. Comprising the fundamental frequency of the signal, the first significant peak at a positive frequency corresponds to the bit rate. The higher harmonics which arise from the sharp transitions in Unipolar RZ show up as the smaller peaks occurring at equally spaced frequency intervals. Unipolar RZ's square-like form comprises several sinusoidal components at various frequencies, hence several harmonics are produced. The fundamental frequency determines the distance between the peaks; the height of the peaks reveals the power carried at every frequency.

Plot 4: Semilog Plot—Log PSD vs TfTf

To underline lesser frequency components, this graph shows the PSD on a logarithmic scale. The regularly spaced peaks verify the existence of harmonics, in which the highest peak corresponds to the fundamental frequency and the next peaks are its harmonic equivalents. Higher harmonics carry less power, hence most of the energy in the signal is held at the lower frequencies as shown by the declining height of the peaks. RZ spreads its power more evenly over several frequencies, so it is more bandwidth-efficient than NRZ, whose power was more concentrated at low frequencies and DC.

Unipolar RZ has a periodic structure, hence the peaks in the PSD graphs show regular frequency intervals. Its Fourier Transform includes harmonic of the fundamental frequency. Unipolar RZ had less DC energy and more evenly distributed frequency components than Unipolar NRZ, which featured a significant DC peak. Multiple peak at harmonics indicate that the signal comprises of sinusoidal components at different frequencies, thereby improving bandwidth efficiency than NRZ. Still lacking negative values, though, it is not as spectrally balanced as Polar RZ.

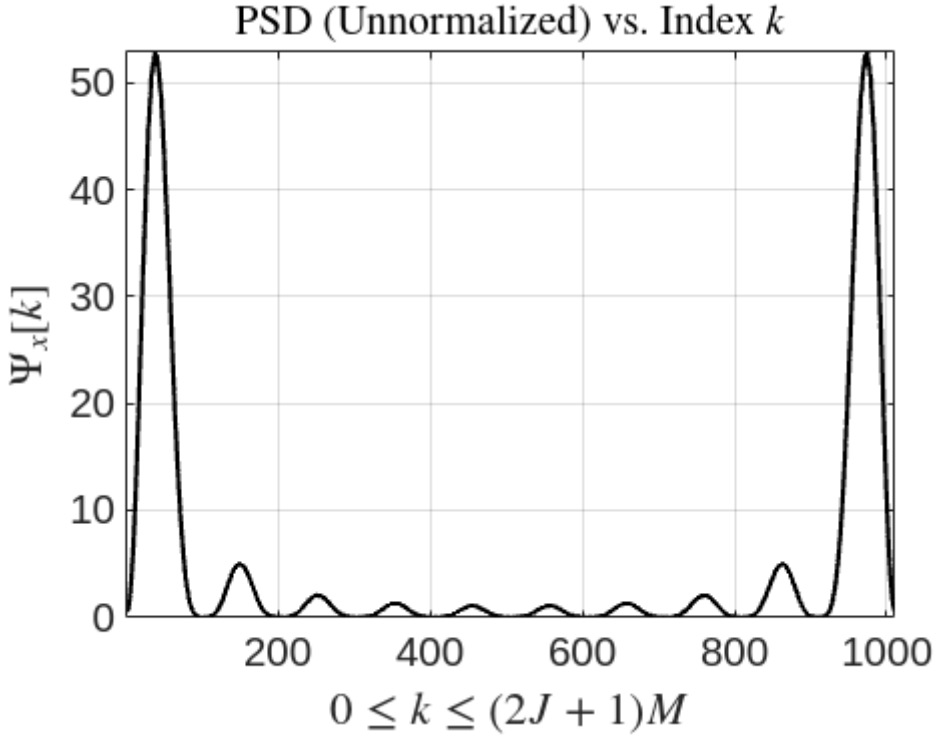
Task d)

Code:

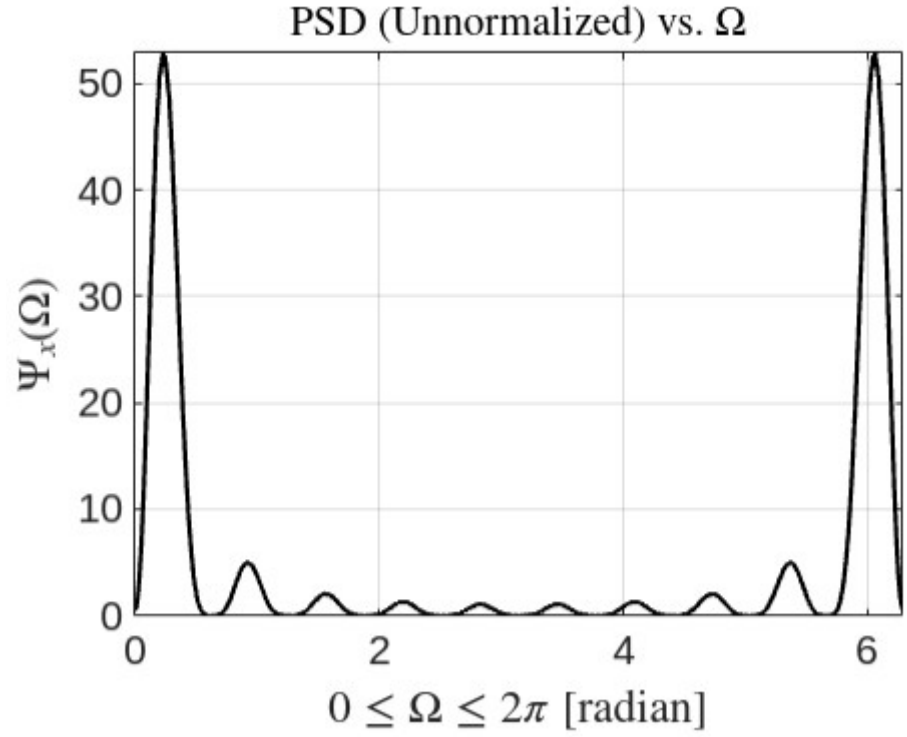
```
%task d:
clc; clear; close all;
A = 1; J = 50; M = 10; L = 20000;
PSD = 0; T = 1; Ts = T/M; fs = 1/Ts;

for jj = 1:L
   sNt = [];
   Bits = (sign(randn(1,2*J+1)) + 1) / 2;
   last_polarity = A;
   for ii = 1:2*J+1
      if Bits(ii) == 1
         sNt = [last_polarity * ones(1,M), sNt];
         last_polarity = -last_polarity;
      else
         sNt = [zeros(1,M), sNt];
      end
   end
   PSD = PSD + abs(fft(sNt)).^2;
end

PSD = abs(PSD) / L / (2*J+1) / T;
LPSD = length(PSD);

% Plot 1: Unnormalized PSD vs. Index k
figure(1);
```

```
plot(PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$0\leq k\leq (2J+1)M$', 'FontSize', 16, 'Interpreter', 'latex');
title('PSD (Unnormalized) vs. Index $k$', 'FontSize', 14, 'Interpreter', 'latex');
ylabel('$\Psi_x[k]$', 'FontSize', 16, 'Interpreter', 'latex');

% Plot 2: PSD vs. Normalized Frequency (DTFT-based)
DTFT_xaxis = linspace(0, 2*pi, LPSD);
figure(2);
plot(DTFT_xaxis, PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$0 \leq \Omega \leq 2\pi$ [radian]', 'FontSize', 16, 'Interpreter', 'latex');
title('PSD (Unnormalized) vs. $\Omega$', 'FontSize', 14, 'Interpreter', 'latex');
ylabel('$\Psi_x(\Omega)$', 'FontSize', 16, 'Interpreter', 'latex');

% Plot 3: Normalized and Two-Sided PSD (CTFT-based)
fT = linspace(-T*fs/2, T*fs/2, LPSD);  % Normalized frequency axis
Area = sum(PSD) * (fT(2) - fT(1));
CTFT_PSD = PSD / Area;
CTFT_PSD = [fliplr(CTFT_PSD(end:-1:floor(LPSD/2))), CTFT_PSD(1:floor(LPSD/2)-1)];

figure(3);
plot(fT, CTFT_PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$-Tf_s/2 \leq Tf \leq Tf_s/2$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$\Psi_x(f)$', 'FontSize', 16, 'Interpreter', 'latex');
xlim([-fs/2 fs/2]);
title('PSD vs. $Tf$ [Normalized and Two-Sided]', 'FontSize', 14, 'Interpreter', 'latex');

% Plot 4: Normalized Semilog Plot
figure(4);
semilogy(fT, CTFT_PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14);
xlabel('$-Tf_s/2 \leq Tf \leq Tf_s/2$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$\Psi_x(f)$ [dB]', 'FontSize', 16, 'Interpreter', 'latex');
ylim([1e-3 1]);
xlim(T*[-fs/2 fs/2]);
title('Log PSD vs. $Tf$ [Normalized and Two-Sided]', 'FontSize', 14, 'Interpreter', 'latex');
grid on; axis tight;
```

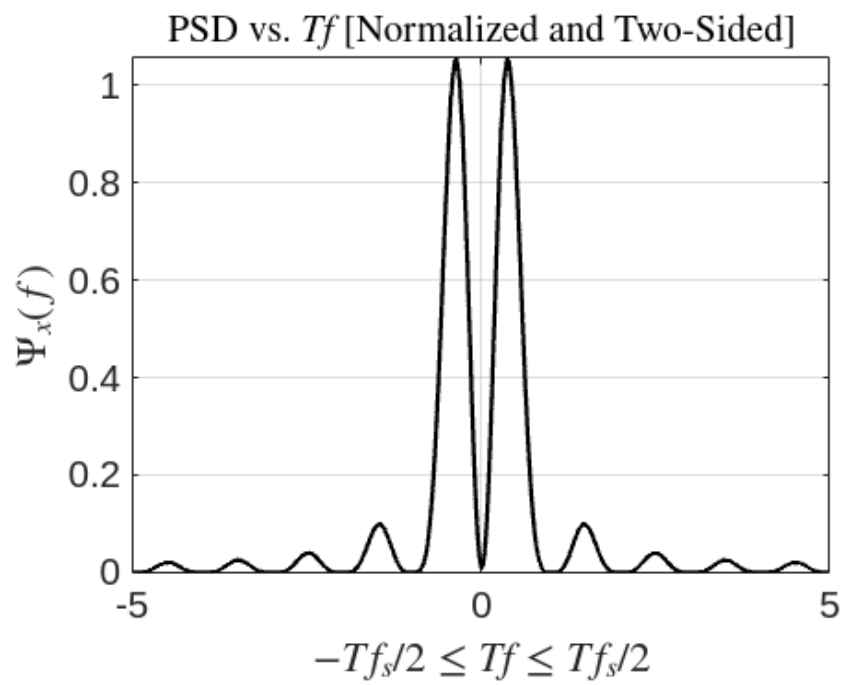
Plot 1 obtained):



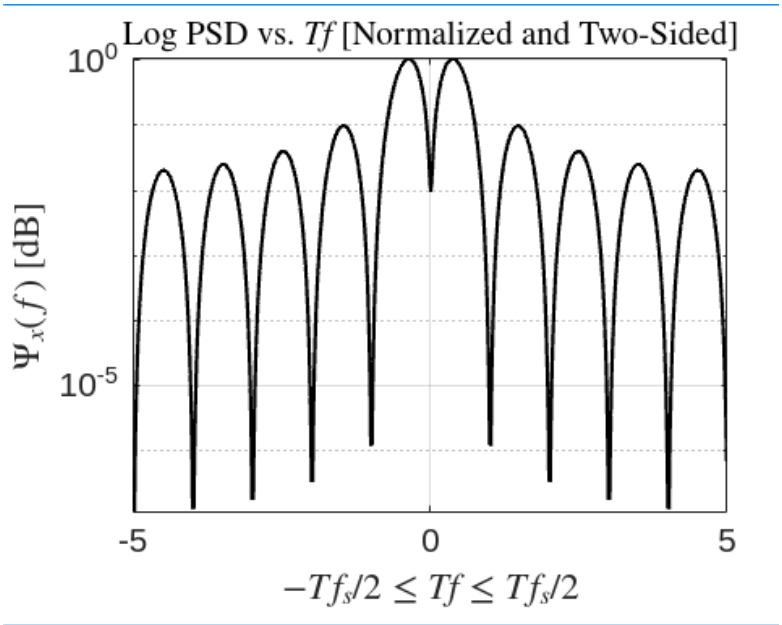
plot 2 obtained)



plot 3 obtained)



plot 4 obtained)



Observation:

In this work, I changed the encoding method to use bipolar NRZ. Whereas a binary "1" in Unipolar NRZ is shown by a fixed high value and "0" by zero, Bipolar NRZ shows "1's with alternating polarity (+A, -A) and "0's remain at zero". Every consecutive "1" bit thus flips polarity to guarantee no net DC component in the signal. The Power Spectral Density (PSD) properties are considerably changed by this modification when compared to earlier line

coding systems.

The software first creates random binary data and codes it using the Bipolar NRZ technique such that every '1' bit is assigned an alternating polarity while '0' stays at zero. The Fourier Transform then converts the resultant waveform into the frequency domain therefore enabling us to examine the distribution of signal power over many frequencies. Particularly in the lower frequency range, Bipolar NRZ acts different from Unipolar NRZ since its DC offset is not consistent.

The first plot represents the PSD against index kk, therefore illustrating the distribution of power among the discrete frequency components of the signal. The periodic character of the alternating "1" bits causes this graph to show several peaks. Bipolar NRZ exhibits no noticeable DC peak, therefore showing that the alternating polarity wipes out any low-frequency energy buildup unlike Unipolar NRZ, where the signal's nonzero average value caused a high peak at k=0 (DC component).

Here we observe comparable periodic peaks in the second plot, the PSD against normalized frequency Ω. The repeated character of the encoded signal causes these peaks to show at particular harmonic frequencies. The lack of power at the lowest frequencies emphasizes even more how Bipolar NRZ removes the DC component, hence more appropriate for transmission over AC-coupled channels where DC bias is unwelcome.

The third graph shows the two-sided PSD against normalized frequency fTfT, therefore offering information on the distribution of power in both positive and negative frequency components. Two prominent peaks here illustrate symmetric concentration of most of the energy in the signal around zero. This results from Bipolar NRZ's alternating structure introducing a pattern whereby energy is equally distributed among positive and negative frequency components.

The logarithmic PSD representation forms the fourth plot and aids in the frequency content over a larger dynamic range visualisation. The alternating character of Bipolar NRZ causes constructive and destructive interference that corresponds with the periodic dips and peaks in this plot. This system spreads energy more effectively over several harmonic levels than Unipolar NRZ, which wastes power at DC.

Unlike Unipolar NRZ, which has a strong DC component due to its nonzero average, Bipolar NRZ lacks a DC component so there is no peak at zero frequency. Because of their alternating character, '1' bits disperse energy more effectively and concentrate it at particular harmonic frequencies. Because it removes needless low-frequency power and stops baseline wander a common problem in long sequences of ones in Unipolar NRZ bipolar NRZ is thus more bandwidth-efficient. Furthermore, this encoding method is more sensible for real-world uses since it is better matched for channels without DC components.

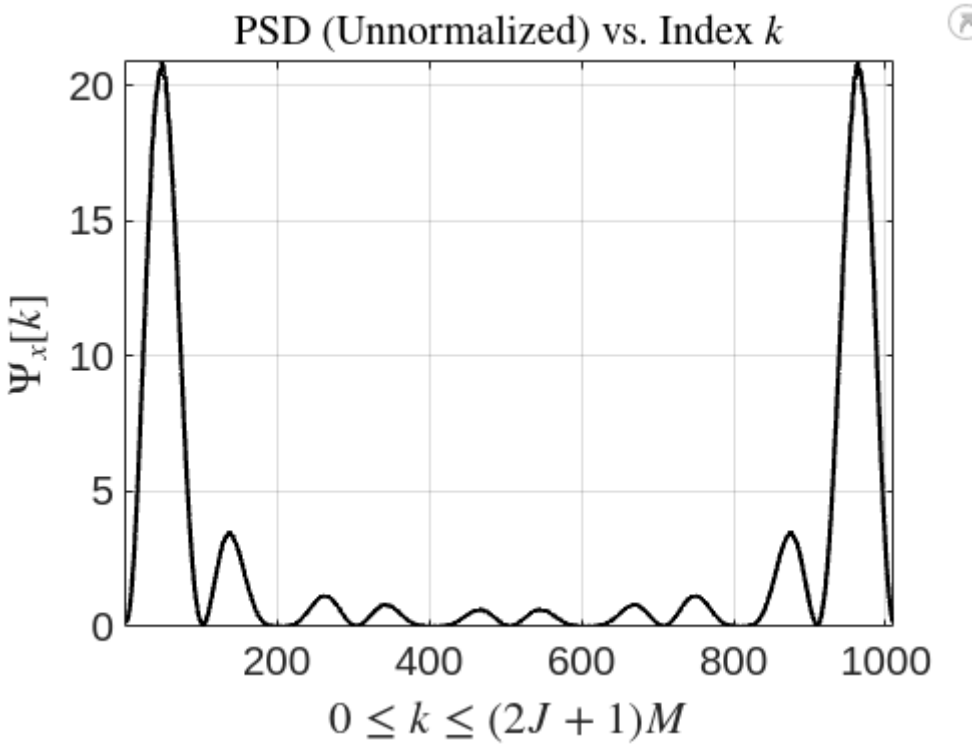
Task e Bipolar-RZ)

code:

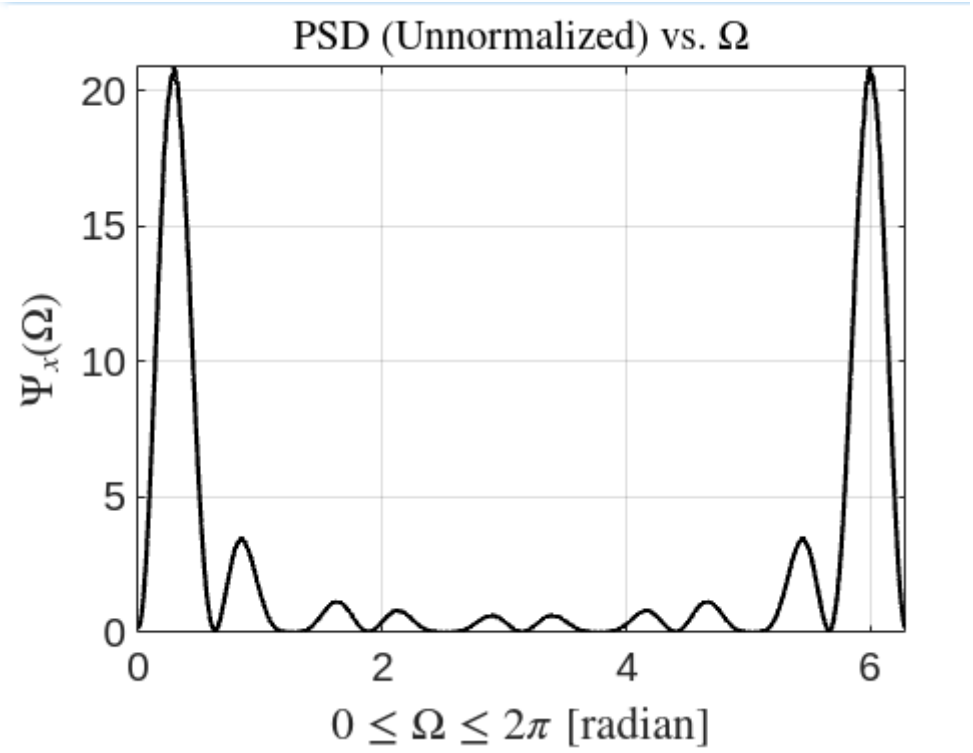
```
%task e:
clc; clear; close all;
A = 1; J = 50; M = 10; L = 20000;
PSD = 0; T = 1; Ts = T/M; fs = 1/Ts;

for jj = 1:L
   sNt = [];
   Bits = (sign(randn(1,2*J+1)) + 1) / 2;
   last_polarity = A;

   for ii = 1:2*J+1
      if Bits(ii) == 1
         sNt = [last_polarity * ones(1,M/2), zeros(1,M/2), sNt];
         last_polarity = -last_polarity;
      else
         sNt = [zeros(1,M), sNt];
      end
   end
   PSD = PSD + abs(fft(sNt)).^2;
end

PSD = abs(PSD) / L / (2*J+1) / T;
LPSD = length(PSD);

% Plot 1: Unnormalized PSD vs. Index k
figure(1);
plot(PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$0\leq k\leq (2J+1)M$', 'FontSize', 16, 'Interpreter', 'latex');
title('PSD (Unnormalized) vs. Index $k$', 'FontSize', 14, 'Interpreter', 'latex');
ylabel('$\Psi_x[k]$', 'FontSize', 16, 'Interpreter', 'latex');

% Plot 2: PSD vs. Normalized Frequency (DTFT-based)
DTFT_xaxis = linspace(0, 2*pi, LPSD);
figure(2);
plot(DTFT_xaxis, PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$0 \leq \Omega \leq 2\pi$ [radian]', 'FontSize', 16, 'Interpreter', 'latex');
title('PSD (Unnormalized) vs. $\Omega$', 'FontSize', 14, 'Interpreter', 'latex');
ylabel('$\Psi_x(\Omega)$', 'FontSize', 16, 'Interpreter', 'latex');

% Plot 3: Normalized and Two-Sided PSD (CTFT-based)
fT = linspace(-T*fs/2, T*fs/2, LPSD);  % Normalized frequency axis
Area = sum(PSD) * (fT(2) - fT(1));
```

```
CTFT_PSD = PSD / Area;
CTFT_PSD = [fliplr(CTFT_PSD(end:-1:floor(LPSD/2))), CTFT_PSD(1:floor(LPSD/2)-1)];

figure(3);
plot(fT, CTFT_PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14); axis tight; grid on;
xlabel('$-Tf_s/2 \leq Tf \leq Tf_s/2$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$\Psi_x(f)$', 'FontSize', 16, 'Interpreter', 'latex');
xlim([-fs/2 fs/2]);
title('PSD vs. $Tf$ [Normalized and Two-Sided]', 'FontSize', 14, 'Interpreter', 'latex');

% Plot 4: Normalized Semilog Plot
figure(4);
semilogy(fT, CTFT_PSD, 'k-', 'linewidth', 1.5);
set(gca, 'FontSize', 14);
xlabel('$-Tf_s/2 \leq Tf \leq Tf_s/2$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$\Psi_x(f)$ [dB]', 'FontSize', 16, 'Interpreter', 'latex');
ylim([1e-3 1]);
xlim(T*[-fs/2 fs/2]);
title('Log PSD vs. $Tf$ [Normalized and Two-Sided]', 'FontSize', 14, 'Interpreter', 'latex');
grid on; axis tight;
```

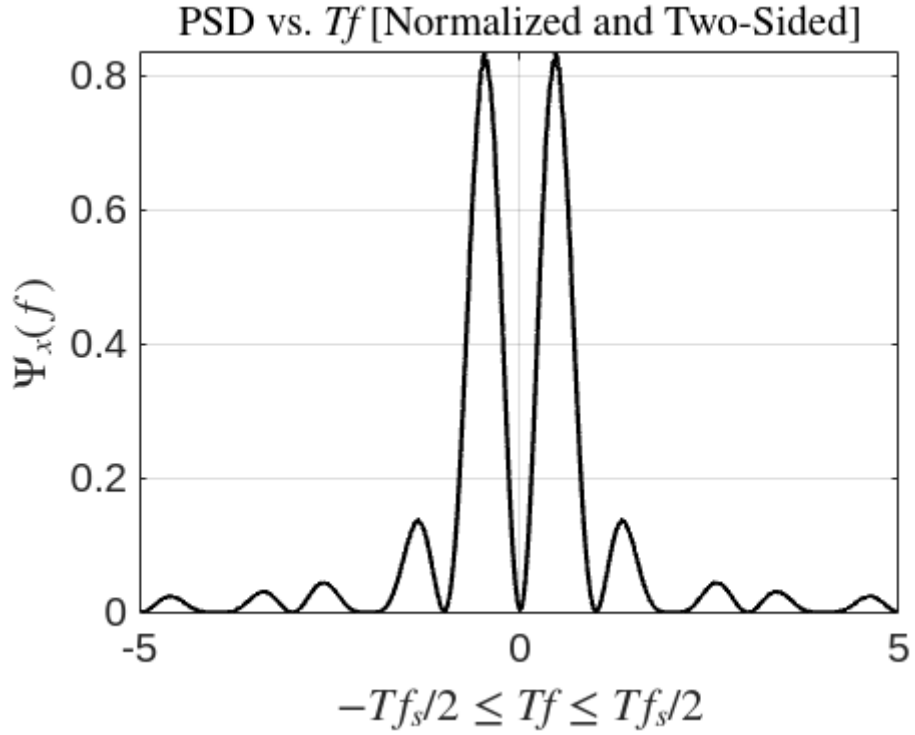
plot 1 obtained)



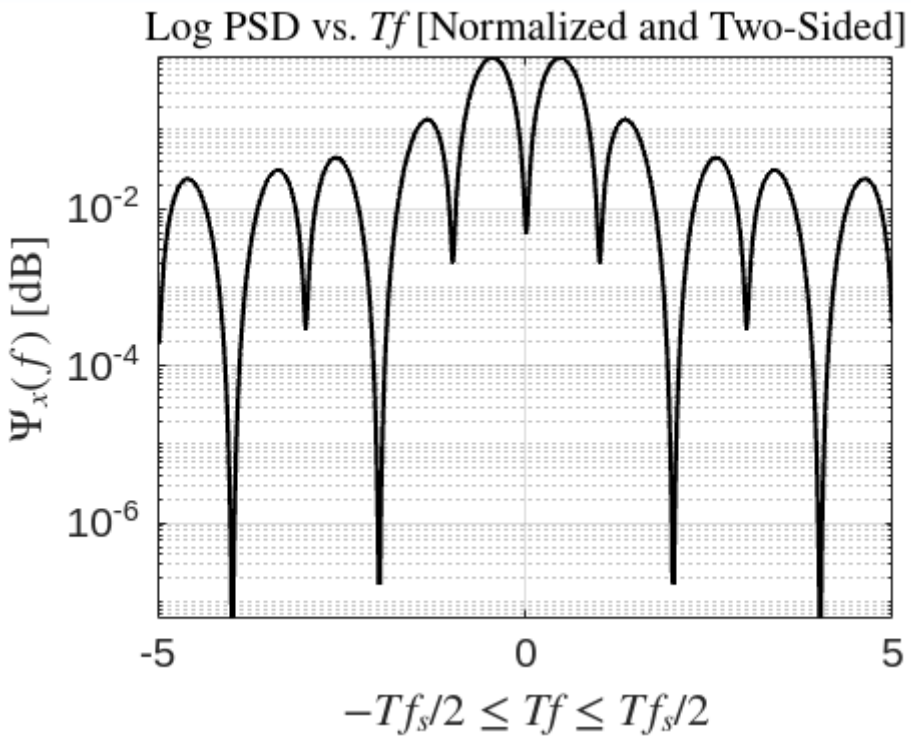
plot 2 obtained)



plot 3 obtained)



plot 4 obtained)



Observation:

I used the Bipolar RZ (Return-to- Zero) approach in doing this work. Every bit "1" in this encoding technique is expressed by alternating values of +A and -A, therefore shifting the signal between positive and negative levels for successive "1" bits. Still, the signal returns to zero in the second half of every bit length unlike the Bipolar NRZ (Non-Return-to- Zero) system. The bit "0" stays zero all during its lifespan. For communication systems, this alternating character of the signal helps to balance the whole power distribution and lessens the existence of a DC bias, therefore increasing their efficiency.

Plot 1, which illustrates the Power Spectral Density (PSD) against index k, revealed two rather significant spectrum edges. These peaks suggest high frequency strong frequency components exist. The slow decline in PSD between these peaks points to more equal distribution of energy over the frequency spectrum than in the Unipolar NRZ system. The total energy is more balanced since bipolar RZ has both positive and negative values alternating, therefore lowering the dominance of low-frequency components.

Plot 2 plots the PSD against normalized frequency Ω. Along with ripples in between, the spectrum displays clear peaks at the beginning and end. Periodic character of the Bipolar RZ signal causes these ripples. Every bit "1" cycles between +A and -A, therefore introducing harmonics at several frequencies that produce these patterns in the frequency domain. The energy is more distributed in comparison to Unipolar NRZ, so limiting the

buildup of too much energy at lower frequencies.

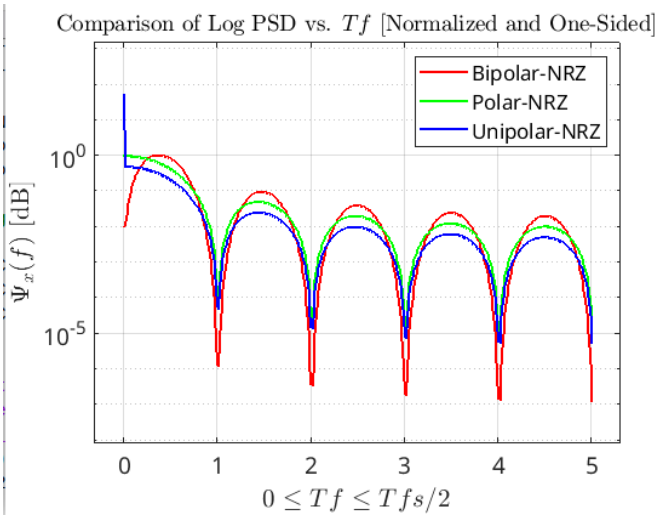
Plot 3 shows the normalized two-sided PSD; the primary peak around zero frequency is smaller than in Unipolar NRZ, thereby verifying that Bipolar RZ has a lowered DC component. At zero frequency, the net DC bias is almost zero since the signal alternates polarity, so there is quite little power. Multiple smaller peaks indicate that the signal spreads its energy throughout a spectrum of frequencies, therefore enhancing spectral efficiency and reducing the problems with DC bias.

Plot 4's PSD shows on a logarithmic scale, which facilitates the side lobe viewing. Where energy is distributed over several harmonics rather than concentrated at a single frequency, the oscillations in the PSD reflect the periodic structure of the Bipolar RZ signal. This energy distribution lowers interference and strengthens the signal for transmission. Better spectrum efficiency is further confirmed by the side lobes' smoother decline than that of Unipolar NRZ.

Multiple benefits exist from bipolar RZ encoding over unipolar NRZ. The signal guarantees absence of long-term DC component since it alternates polarity for every "1" bit. This is significant since in transmission a high DC component might lead to signal deterioration. The return-to- zero characteristic aids in low-frequency energy reduction, hence optimizing the spectrum's efficiency and compactness. Furthermore, the signal is less prone to interference and distortion since the energy is distributed over a greater frequency spectrum.

Bipolar RZ is more appropriate for uses when lowering DC bias is crucial, like in communication channels without low-frequency transmission, than Unipolar RZ. In some communication systems, Bipolar RZ is a recommended choice since the enhanced spectral efficiency guarantees efficient use of additional bandwidth.

Task f) Compare plots of normalized one-sided semilog Ψx (f) of Bipolar-NRZ code, Polar-NRZ code, and Unipolar-NRZ code versus 0 ≤ T f ≤ T fs /2.



Task F sought to evaluate across the range 0≤Tf≤Tfs/2 the normalized one-sided semilog Power Spectral Density (PSD) for Bipolar-NRZ, Polar-NRZ, and Unipolar-NRZ encoding

schemes. Understanding bandwidth efficiency and spectrum features depends on our ability to examine how each encoding method distributes its power over various frequencies, which the depicted PSD enables.

Whereas the Y-axis shows the logarithm of the normalized Power Spectral Density ΨxΨx (f) in dB, the X-axis of the graph shows the normalized frequency TfTf. Each of the three graph curves—bipolar-NRZ (red curve), polar-NRZ (green curve), and unipolar-NRZ (blue curve)—show distinct spectrum features.

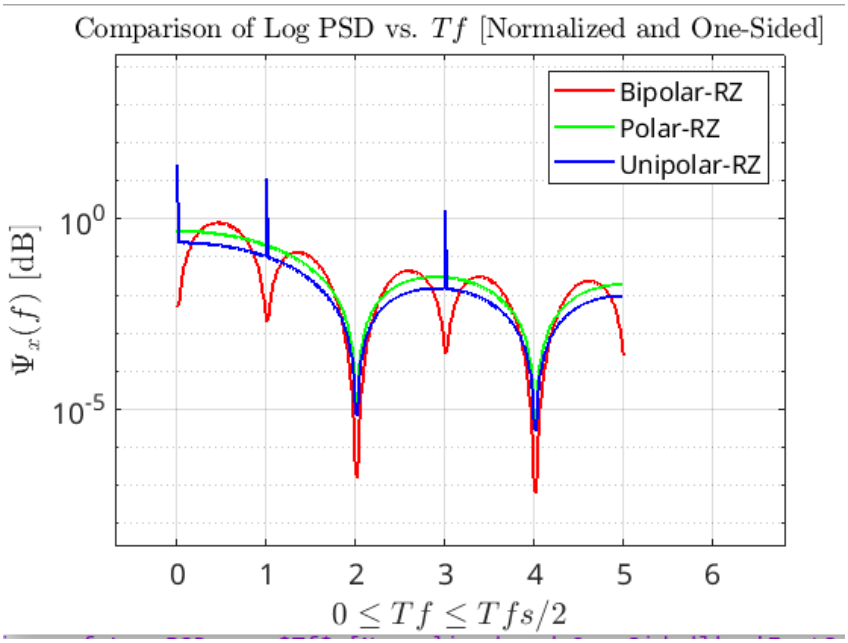
The behavior at zero frequency (DC component) of the graph is among its most obvious features. Strong DC component presence is indicated by the Unipolar-NRZ (blue curve) with peak magnitude at Tf=0Tf=0. Unipolar-NRZ only employs positive values (+A+A for 1 and 0 for 0), thereby producing a nonzero average signal value. With its balanced bit (+A for 1 and -A for 0), the Polar-NRZ (green curve) shows a lesser DC component, therefore lowering the total DC bias. Whereas each "1" bit alternates between ±A±A, therefore balancing out the DC content over time, Bipolar-NRZ (red curve) has essentially no DC component.

A further important finding is the frequency spectrum's distribution of power. With less power sent to high frequencies, bipolar-NRZ encoding shows the most effective power distribution and so is more bandwidth-efficient than the other two techniques. Though it distributes electricity over a somewhat wider bandwidth, Polar-NRZ trails closely. Unipolar-NRZ, on the other hand, is least bandwidth-efficient and generates higher power at higher frequencies, hence perhaps increasing inter-symbol interference (ISI) in communication systems. Its frequency spread is also the biggest.

The PSD's periodic nulls show the frequency intervals where power is leastened. Following a sinc-squared-like decay pattern, the PSD curves exhibit periodic power decreases. With the most sharp nulls, bipolar-NRZ shows more focused spectral power, hence lowering needless frequency components. While Unipolar-NRZ has the slowest decay and hence retains more power at higher frequencies, Polar-NRZ has somewhat more broad peaks and is therefore less ideal for bandwidth-efficient transmission.

Finally, since Bipolar-NRZ reduces DC bias, distributes power efficiently, and preserves a compact spectral form, it is the most bandwidth-efficient technique. Polar-NRZ has somewhat broader spectral dispersion and moderate bandwidth efficiency including some DC component. With a strong DC bias, spreads power over a larger bandwidth, and preserves notable high-frequency components, which increases distortion and ISI susceptibility, unpolar-NRZ is the least efficient. Bipolar-NRZ is thereby recommended in practical communication systems when DC-free transmission and bandwidth efficiency are needed.

Task g)



Using a logarithmic scale, we investigated in this work the Power Spectral Density (PSD) of three distinct Return-to- Zero (RZ) encoding systems: Bipolar-RZ, Polar-RZ, and Unipolar-RZ. The shown graph shows the log PSD in dB throughout the one-sided normalized frequency range 0≤Tf≤Tfs/20≤Tf≤Tfs/2. Examining the curves, we find in all three cases a sinc-like pattern with clear nulls (dips) appearing at particular frequencies; the first of which manifests around Tf=2Tf=2. Where the largest power concentration exists—Tf=0Tf=0—the primary lobe is centered.

Among the three encoding techniques, Bipolar-RZ—shown in red—has the lowest PSD at higher frequencies, therefore indicating improved bandwidth efficiency. The deep nulls imply that it more evenly spreads power over the frequency range. This is so because bipolar encoding switches polarity, therefore minimising the DC component and hence the low-frequency power concentration required. Thanks to its nonzero DC component, Polar-RZ—shown by the green curve—showcases a somewhat greater PSD at low frequencies than Bipolar-RZ. Though less bandwidth-efficient than Bipolar-RZ, it maintains a comparable null structure. Displaying blue, Unipolar-RZ has the largest PSD at low frequencies, so it has a strong DC component and absorbs much more bandwidth. Among the three methods in terms of spectral use, this is the least efficient since extra harmonics distribute energy over a wider frequency range.

Analyzing the PSD's peaks and nulls, we find that the dominant frequency components match the peaks while the nulls show frequencies where destructive interference cancels out power. The most bandwidth-efficient bipolar-RZ shows the fastest degradation in high-frequency content. Conversely, Unipolar-RZ has the slowest decline which causes rising bandwidth use. Polar-RZ suffers a clear DC shift yet balances power concentration between the two.

Practically speaking, Bipolar-RZ is chosen over Unipolar-RZ since it reduces DC shifting-related problems and bandwidth use. While Polar-RZ remains a good choice with moderate spectral efficiency, the energy distribution in Unipolar-RZ makes it less appropriate for power-efficient

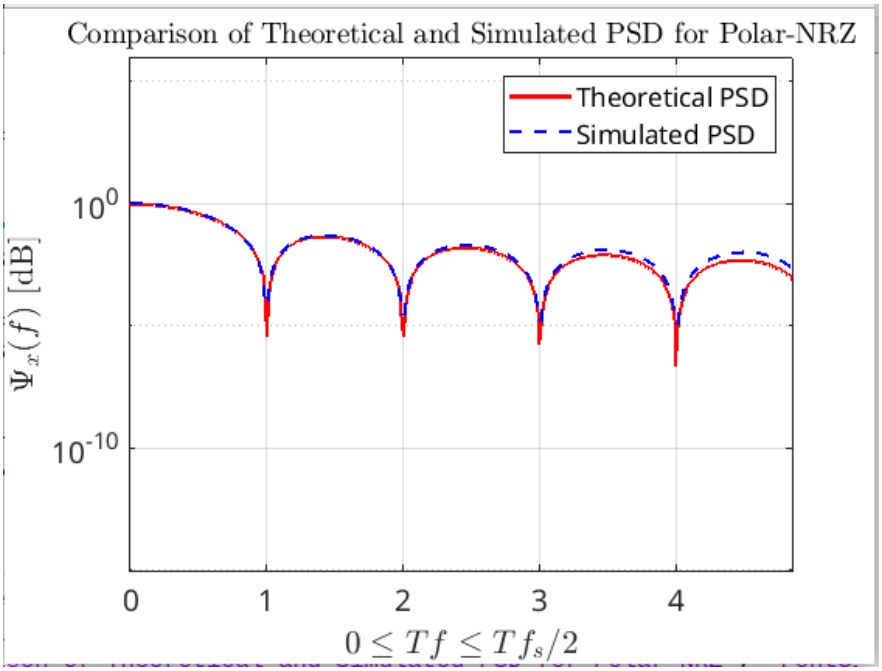
devices. The study reveals how various encoding methods affect frequency properties, therefore proving their relevance for several uses in digital communication.

Task h)

Code:

```
clc; clear; close all;

A = 1;
J = 50;
M = 10;
L = 20000;
T = 1;
Ts = T/M;
fs = 1/Ts;
LPSD = (2*J+1)*M;


PSD_PolarNRZ = 0;

for jj = 1:L
   sNt_PolarNRZ = [];

   Bits = (sign(randn(1, 2*J+1)));

   for ii = 1:2*J+1
      sNt_PolarNRZ = [Bits(ii) * A * ones(1,M), sNt_PolarNRZ];
   end

   PSD_PolarNRZ = PSD_PolarNRZ + abs(fft(sNt_PolarNRZ)).^2;
end

PSD_PolarNRZ = abs(PSD_PolarNRZ) / L / (2*J+1) / T;

fT = linspace(0, T*fs/2, LPSD/2);
%theoritical PSD:
Theoretical_PSD_PolarNRZ = (A^2 * T) .* (sinc(fT * T)).^2;

PSD_PolarNRZ = PSD_PolarNRZ(1:LPSD/2) / max(PSD_PolarNRZ);
Theoretical_PSD_PolarNRZ = Theoretical_PSD_PolarNRZ / max(Theoretical_PSD_PolarNRZ);

figure;
semilogy(fT, Theoretical_PSD_PolarNRZ, 'r-', 'linewidth', 2); hold on;
semilogy(fT, PSD_PolarNRZ, 'b--', 'linewidth', 1.5);
hold off;

set(gca, 'FontSize', 14);
```

```
xlabel('$0 \leq Tf \leq T f_s/2$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$\Psi_x(f)$ [dB]', 'FontSize', 16, 'Interpreter', 'latex');
title('Comparison of Theoretical and Simulated PSD for Polar-NRZ', 'FontSize', 14, 'Interpreter',
'latex');
legend('Theoretical PSD', 'Simulated PSD');
grid on; axis tight;
```



To depict the Polar-NRZ encoding system, first I created a random series of binary data including +A and -A values. To guarantee appropriate signal representation, I then built the time-domain waveform assigning each bit an MM sample length. I subsequently calculated the Power Spectral Density (PSD) of the simulated signal using the Fast Fourier Transform (FFT). I ran this over 20,000 times and averaged the outcomes to get a steady approximation. The predicted spectral distribution of the Polar-NRZ signal is then determined theoretically by means of the formula Ψx(f)=A2T·sinc2(fT). Next, on a semilogarithmic scale, I drew the theoretical and simulated PSDs; the blue dashed line indicated the simulated PSD and the red solid line indicated the theoretical PSD. With a primary lobe centered at zero frequency and side lobes declining at higher frequencies, I finally examined the findings to determine that the simulated PSD closely fits the expected sinc-squared form, therefore supporting the theoretical predictions of Polar-NRZ spectral features.

Final conclusion from the output plot:

The major lobe of the plot has zero frequency, which corresponds to the dominant low-frequency energy of the signal, and nulls occurring at integer multiples of 1/T1/T, therefore verifying the characteristic sinc-squared form of the Polar-NRZ spectrum. With declining strength at higher frequencies, the side lobes show that most of the signal energy is focused at lower frequencies. This trend confirms that Polar-NRZ has a somewhat low spectral efficiency and uses more bandwidth since its rectangular pulse form produces slower frequency domain decay. The exact match between

the simulated and theoretical curves shows that the used simulation technique appropriately catches the spectral characteristics of the Polar-NRZ line coding scheme.

Activity 3)

Task 1)Unipolar codes include DC voltage in their power spectral density (PSD). What are the potential drawbacks of DC voltage in data transmission?

Unipolar codes include a nonzero DC component in the frequency domain, so their Power Spectral Density (PSD) includes DC voltage and can lead to several issues in data transmission. One main disadvantage of the DC component is poor power consumption as, although it consumes power, it does not help to transfer essential information, therefore causing needless energy loss, particularly in battery-powered devices and long-distance communication systems where power efficiency is vital. Commonly used in communication systems including telephone lines, optical fiber networks, and radio transmission, transformers and capacitors also block DC signals, so attenuating or totally lost signals with strong DC components can cause data corruption or transmission failures.

Low-frequency interference is another problem whereby low-frequency signals can be distorted and signal overlap results from the DC component at zero frequency, therefore complicating data recovery. Furthermore, baseline wander in digital transmission results from the DC component causing the average voltage level to change over time, which makes it challenging for the receiver to properly read binary levels (0s and 1s), hence increasing bit mistakes and signal recognition problems. Moreover, a DC bias disturbs clock recovery circuits, which makes it more difficult for the receiver to reach ideal synchronisation, hence causing data decoding mistakes.

DC voltage causes unequal power distribution in wireless and radio frequency (RF) communication, therefore focusing signal energy at zero frequency is useless for transmission and causes ineffective bandwidth use. Often used instead of unipolar coding to avoid these problems are bipolar and polar coding methods, which guarantee a balanced number of positive and negative values thereby eliminating the DC component and hence enhancing transmission efficiency. To limit the detrimental impacts of DC bias, other methods including Manchester encoding, differential coding, and DC blocking filters are also applied so guaranteeing improved data integrity and system performance in useful communication channels.

Task 2) **What advantages does RZ have compared to NRZ? What disadvantages does RZ have compared to NRZ?**

Two often used line coding systems with different advantages and drawbacks are Return-to- Zero (RZ) and Non-Return-to- Zero (NRZ). RZ encoding offers one of the main benefits in that it introduces a transition to zero in the midst of each bit period, therefore improving synchronisation between the transmitter and receiver. This change helps preserve timing precision, hence lowering the bit misinterpretation risk—a particularly helpful benefit in situations needing exact timing recovery. Furthermore, RZ encoding is more appropriate for AC-coupled transmission lines and helps to lower DC signal accumulation since it usually minimizes low-frequency components in the signal, especially in bipolar and polar schemes.

Still, RZ encoding has major disadvantages relative to NRZ. Its greater bandwidth need is one main

drawback since every bit experiences an extra transition, therefore doubling the frequency content of the signal. In systems running under bandwidth restrictions, this higher demand for bandwidth can restrict efficiency in communication. Moreover, RZ signals eat more energy per bit than NRZ since they need extra transitions every bit. The mid-bit transition means that energy is consumed not only to encode the bit but also to return to zero, hence increasing the power consumption. Furthermore, as RZ encoding employs three different levels (+A, 0, and -A) in bipolar systems, differentiating between these levels demands more energy at the receiver to maintain signal integrity and guarantee appropriate detection, especially in noisy surroundings.

On high-data-rate transmissions, where conserving power and bandwidth is vital, NRZ encoding is more energy-efficient and requires less bandwidth, so preferred. But because a bit period lacks natural transitions, especially when sending extended sequences of identical bits, NRZ signals—especially NRZ-Level—are more prone to synchronizing problems. Lack of transitions makes it more difficult for the receiver to derive time information, hence raising bit error risk.

System requirements ultimately determine which of RZ and NRZ to use; they balance the trade-offs between bandwidth efficiency, synchronizing needs, power consumption, and system complexity. RZ may be a preferable option for enhanced synchronizing and low DC components if a system can accept increased power consumption and has lots of bandwidth. NRZ would be the better choice, nevertheless, if bandwidth saving and energy efficiency are absolutely vital.

Task 3) Bipolar codes, unlike polar codes, exhibit a spectral null at f = 0. Is this an advantage or a disadvantage? Briefly explain your opinion.

Bipolar coding techniques, such as Alternate Mark Inversion (AMI), display a spectral null at f=0f=0, which indicates that the signal does not contain any negative direct current (DC) components. It is generally agreed upon that this quality is advantageous in digital communication systems for a number of different reasons. Bipolar signals are compatible with AC-coupled transmission lines since they do not have a DC component. This is because AC-coupled transmission lines filter out DC levels. By doing so, the signal is guaranteed to be able to travel via a variety of transmission media without being distorted. In addition, the removal of the DC component helps to lessen signal droop, which is a progressive drop in amplitude over time. This helps to improve the integrity of the signal during transmission. Due to the fact that it adheres to a precise alternating voltage pattern, bipolar coding also has the advantage of contributing to improved error detection. When this pattern is broken, faults are able to be identified with relative ease. The fact that bipolar signals have regular transitions, which reduces the chance of timing mismatches, is another reason why they provide superior synchronization between the transmitter and the receiver.

However, despite the fact that the spectral null at f=0f=0 is advantageous, bipolar coding may experience difficulties when extremely long sequences of bits that are identical are communicated. Unfortunately, this can result in a loss of synchronization because there are not enough transitions. For the purpose of addressing this problem, coding strategies such as B8ZS (Bipolar with 8-Zero Substitution) and HDB3 (High-Density Bipolar 3) are implemented. These strategies involve the intentional violation or substitution of synchronization information.

As a whole, the spectral null at f=0f=0 is an advantage in bipolar codes because it improves error detection, decreases signal distortion, enhances transmission reliability, and ensures synchronization. As a result, it is a preferred choice for a great deal of digital communication systems.

Task 4)

task 4)

Prove that

$$\sum_{l=-\infty}^{\infty} e^{-jl\omega T} = \frac{1}{T} \sum_{l=-\infty}^{\infty} \delta\left(f - \frac{l}{T}\right)$$

⇓

this equation represents the Poisson Summation formula.

the given LHS

$$\sum_{l=-\infty}^{\infty} e^{-jl\omega T}$$

this is actually an infinite sum of complex exponentials

this shows an impulse train in the frequency domain which is known as "Dirac comb function"

$$\sum_{l=-\infty}^{\infty} e^{-jl\omega T} = 2\pi \sum_{l=-\infty}^{\infty} \delta\left(2\pi f - \frac{2\pi l}{T}\right)$$

dividing both sides by 2π

$$\Downarrow$$

$$\sum_{l=-\infty}^{\infty} e^{-jl\omega T} = \frac{1}{T} \sum_{l=-\infty}^{\infty} \delta\left(f - \frac{l}{T}\right)$$

Hence Proved //.

$$\because \delta(2\pi u) = \frac{1}{2\pi}\delta(u)$$

$$\Downarrow$$

then we get

6/7

reference link by which I saw comb function and shah property :

https://en.wikipedia.org/wiki/Dirac_comb#Fourier_series

Task 5) Explain why it is possible to obtain Ψx (f) numerically from Ψx [k] multiplied with Ts2 .

In signal processing, the Power Spectral Density (PSD) characterizes the distribution of power of a signal over several frequencies. Denoted as Ψx(f), the PSD of a continuous-time signal offers understanding of its frequency components. Ψx(f). Actually signals are often sampled, producing a discrete-time representation of the signal. Computed using the Discrete Fourier Transform (DFT) or its effective implementation, the Fast Fourier Transform (FFT), the PSD of this discrete-time signal is expressed as Ψx[k].

It is necessary to consider the effects of sampling if one is to precisely connect the discrete-time PSD Ψx[k]Ψx(f) to the continuous-time PSD Ψx(f). The discrete sequence x[n]=x(nTs)x[n]=x(nTs´) is obtained when a continuous-time signal x(t) is regularly sampled at Ts. A periodic frequency spectrum results from spectral replicas (aliasing) at integer multiples of the sampling frequency fs=1/Tsfs, produced by this sampling procedure. The link between the continuous- and discrete-time Fourier transforms reveals periodic repeating spectral components of the sampled signal.

One needs a scaling factor of Ts2 **to move from discrete-time frequency representation to a continuous one.** This scaling guarantees that, in terms of power per unit frequency, the power spectral density keeps the proper physical meaning. A scaling factor proportional to the square of the sample interval, Ts2, must be used **to compress the frequency spectrum** and **preserve the appropriate amplitude of the spectral components by sampling a continuous signal** at intervals of Ts. This guarantees the frequency characteristics of the signal by precisely matching the energy shown in the discrete domain to the energy in the continuous domain.

This change is required to maintain energy conservation as Parseval's theorem describes that the total power of a signal in the time domain must match the total power in the frequency domain. FFT computes the spectrum in discrete frequency bins; so, every frequency bin represents a small portion of continuous frequencies. **Multipling by Ts2 guarantees that the computed PSD preserves the proper energy distribution** when translating this representation back into the continuous-time domain.

Furthermore, this change guarantees that the estimated PSD corresponds with theoretical expectations. Discrete PSD Ψx[k] is computed from sampled signals and is normalized depending on the sampling period and sample count. When switching to the continuous-time representation where power is measured per unit frequency instead of each discrete frequency index this normalizing has to be changed. **By means of Ts2, this unqeuality is accounted for and guarantees that the computed PSD fairly reflects the spectral distribution of the continuous signal.**

Correct mapping the discrete power spectral density to the continuous power spectral density depends on multiplying Ψx[k]Ψx[k] by Ts2Ts2. This change guarantees correct distribution of energy over frequencies, preserves consistency with the continuous-time Fourier transform (CTFT), and enables exact numerical computation of Ψx(f)Ψx(f). Where exact spectral analysis is essential, this method is extensively applied in digital signal processing for uses in communications, radar, picture processing, and audio signal analysis.

Task 6) Unipolar and Bipolar codes are commonly thought to cause synchronization issues. What is your understanding of this?

Line coding is the process of converting digital data into a form fit for transmission across a communication channel. This process functions within the domain of digital communication. Two usually used coding techniques are bipolar and unipolar encoding. Every one of these encoding methods has advantages and disadvantages on its own. One of the most important problems different coding methods offer is the phenomena known as synchronizing loss, which results from the receiver starting to lose track of the timing of arriving bits. Most of the time, this issue shows up when there are long stretches of like bits mostly consecutive 0s that cause the signal to lack transitions.

The Unipolar encoding scheme presents the binary data utilizing two separate voltage levels. The binary "1" uses a high voltage (+V), and the binary "0" uses a zero value (0V). This basic method introduces a major synchronizing problem even if it is easy to apply. When the data consists of a long sequence of consecutive zeros, the signal will remain at 0V for a long amount of time, hence producing a lack of transitions. These transitions are definitely required if the receiver is to stay in time with the sender's clock. In the lack of transitions, the receiver cannot distinguish the beginning and finish of each bit, hence the interpretation of the data may be erroneous. By means of laboratory operations, it was shown that Unipolar-NRZ codes possessed a significant DC component at 0 Hz. This element helps to cause synchronizing loss rather than carrying any significant data. Unipolar encoding shows a significant quantity of low-frequency power based on the results of the Power Spectral Density (PSD) research, which increases its sensitivity to synchronizing issues.

Often called Alternate Mark Inversion (AMI), bipolar encoding makes use of three voltage levels: +V, -V, and 0V. Conversely, a binary "0" is always expressed by 0V a binary "1" can take on either a positive or negative voltage. Apart from making sure every binary '1' generates a voltage transition, which facilitates synchronisation, this alternating pattern is beneficial since it reduces the DC component of the signal. Conversely, if the sent data consists of long stretches of zeros, the signal will stay at 0V for a really long period similar to the unipolar encoding. Lack of transitions resulting from this causes a loss of synchronisation. Two such advanced techniques created to solve this problem are bipolar 8-zero substitution (B8ZS) and high-density bipolar of order 3 (HDB3). These techniques include the alteration of long sequences of zeros by means of simulated transitions. This ensures that the receiver follows the broadcaster exactly.

Lab experiments showed me that bipolar codes show a spectral null at f = 0, meaning they do not suffer from too strong DC components in the same manner unipolar codes do. This is a major benefit since it eliminates the unnecessary loss of low frequency power. Notwithstanding this advantage, bipolar codes still suffer with synchronisation when extended sequences of zeros arise. Zero-substitution methods like B8ZS and HDB3 are hence needed to guarantee dependable transmission.

In short, one might say that both unipolar and bipolar encoding schemes can suffer with synchronizing problems. This is because transitions might not happen in long runs of like bits. Unipolar encoding has major synchronizing issues since it lacks transitions for long sequences of zeros and has a constant DC component. With its alternating voltage pattern, bipolar encoding helps to reduce some of these problems; however, it requires the use of additional methods to control long sequences of zeros. The PSD study conducted in the lab demonstrated how different line coding approaches distribute power across frequencies, therefore proving these effects. Some of the contemporary communication methods applied to solve synchronizing problems are Manchester encoding, scrambling techniques, and clock recovery circuits. These techniques guarantee dependable data flow in high-speed networks like optical fiber systems, Ethernet, and mobile networks.

Task 7) What is your understanding of Dicode NRZ? Briefly describe how it operates.

Digital communication makes use of a three-level line coding scheme called Dicode NRZ (Non-Return-to- Zero) to efficiently transfer binary data and concurrently minimise the presence of low-frequency components in the signal. Dicode NRZ presents a third level, zero voltage, which generates a more balanced and effective transmission technique in terms of energy use. Unlike the usual NRZ encoding, which runs with two levels—high for binary "1" and low for binary "0"—this is in contrast. Dicode NRZ is predicated on the basic theory that the signal moves in polarity anytime there is a change between binary "0" and "1". The output will remain at zero voltage, nevertheless, if the bits that follow each other are exactly the same. This suggests that if a "0" changes to a "1" or a "1," the voltage will swing from positive to negative levels. On the other hand, the output voltage will stay at zero if the sequence of the inputs does not change—that is, if there are many consecutive 0s or 1s. This approach greatly reduces DC components—power levels not required and devoid of any significant data. The signal transmissions so are more energy-efficient.

By means of an example, one can grasp the operations of Dicode NRZ. Assume for the moment we forward the binary sequence 011001 to the recipient. In a conventional NRZ configuration, this would mean that high and low voltage would be continuously changing. Conversely, the output behaves differently in Dicode NRZ: it becomes a positive pulse when the first transition from '0' to '1' occurs; it stays zero voltage when the next bit is also '1'. While the sequence runs from "1" to "0," the output voltage moves from being positive to being negative; while it stays at "0," it keeps being at what is known as zero voltage. The polarity will once more be turned to positive if the sequence is reversed from '0' to '1' at a later point of time. This alternating pattern ensures that transitions only happen when the data changes, therefore preserving synchronisation and minimising the non-essential energy loss.

Dicode NRZ offers one of the most important advantages in terms of energy economy. Particularly in cases of vast distance between the two places, signals in communication systems with a significant direct current component need more power to be maintained, which could lead to signal degradation. Dicode NRZ is particularly helpful for optical fiber connections, which are sensitive to signal distortion when low-frequency material levels are too high since it naturally lowers DC levels. Dicode NRZ also finds application in magnetic storage devices such as hard drives, where reducing low-frequency interference helps to prevent data loss and improves device read-through efficiency. In applications requiring fast data transfer, dicode NRZ is a method that helps preserve signal integrity while concurrently reducing power usage. Among other things, Ethernet and serial communication systems are instances of such uses.

Dicode NRZ does, however, have one disadvantage in that it may incur synchronizing loss when sending long sequences of comparable bits—that is, a string of 0s or 1s in a succession. Since the encoding will remain at zero voltage over these sequences, the receiver may find it challenging to sustain proper clock synchronizing throughout them. This is so since track timing will not be tracked by voltage transitions. This is similar to the problem with Unipolar NRZ encoding, whereby long sequences of zeros could cause problems with synchronizing. Often used in order to address this problem are scrambling methods or clock recovery circuits. These are used to ensure that the signal maintains synchronisation by means of enough transitions within it. Other encoding techniques occasionally employed with Dicode NRZ produce frequent transitions in actual communication systems. This assists to guarantee that the signal stays well-structured, thereby raising the possibility of the receiver understanding it exactly.

Finally, Dicode NRZ is a useful technique of line coding that balances the ease of implementation, spectrum efficiency, and energy economy. Because this technique lowers the number of DC components, optical communications, fast data transfer, and magnetic storage systems are all great

uses for it. Over long identical bit sequences, it still suffers with synchronizing problems even though it helps preserve signal quality and economy. Many times, these difficulties are resolved by applying other strategies as adaptive clock recovery techniques or scrambling. Dicode NRZ is still a fundamental aspect of digital transmission systems because of its several advantages, especially in cases when low power consumption and spectrum performance enhancement are required.

Reference link:

www.njit.edu/web.