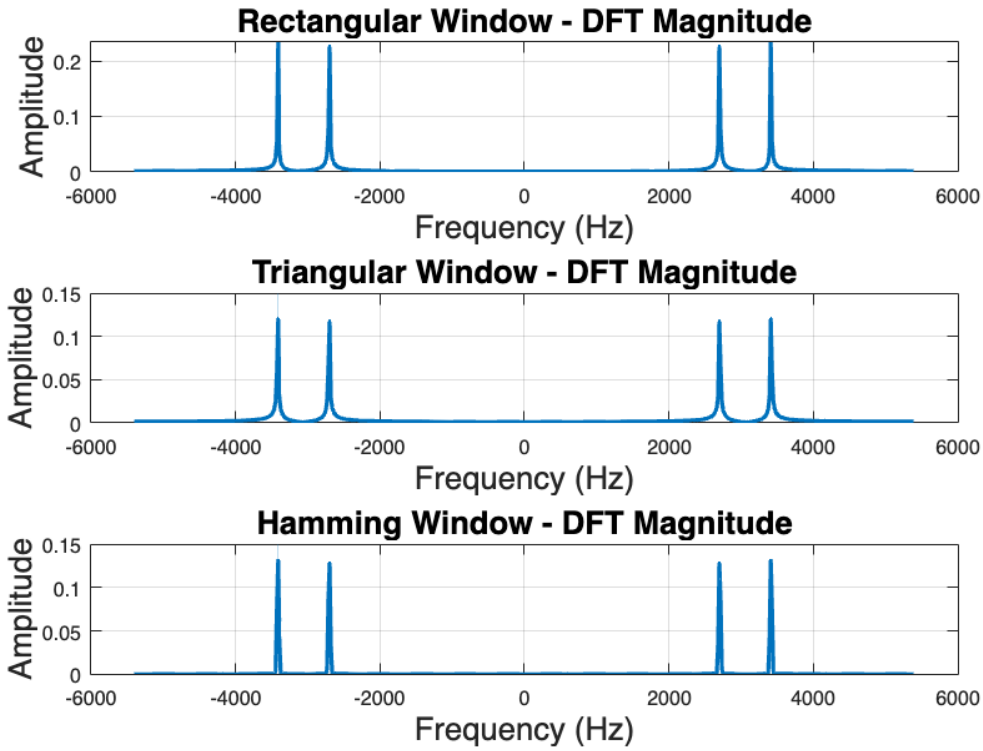
# Presentation of the plots for DFT, DTFT, and CTFT Theoretical and Numerical (Magnitude and Phase for each window)

## DFT

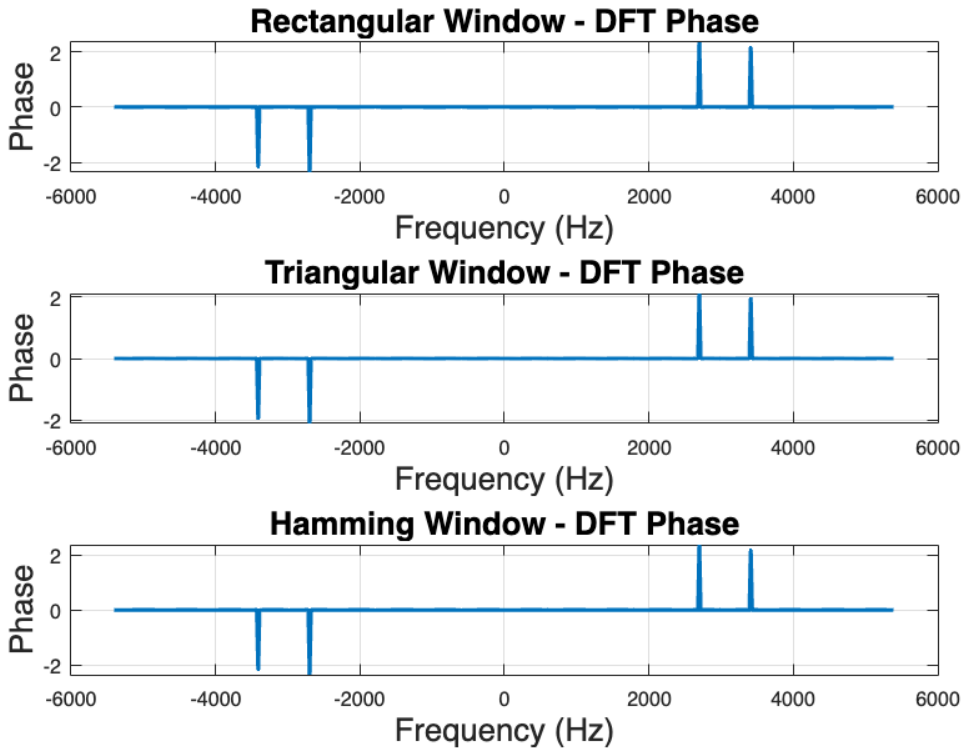
```
clc
clear all
close all
% Message frequency and carrier frequency
f0 = 355.35; % Hz, message frequency
fc = 6.6 * f0; % Hz, carrier frequency
% Sampling frequencies and corresponding time steps
Fs = 4*(f0+fc);
Ts = 1/Fs;
N0 = round(Fs/(f0 + fc));
% factor = [1*N0, 15*N0, 100*N0];
% t = -factor(1) * Ts:Ts: factor(1)* Ts;
% n = -Fs/2:Fs/2;
% n = round(t/Ts);
% N = length(n);
a=20;
N = a*(Fs/(f0));
n=0:N-1;
deltf=Fs/N;
dsbsc_dt = dsbsc(n, fc, f0, Fs);
rect_windowed=1.*dsbsc_dt;
tri_windowed=tri(n, N).*dsbsc_dt;
ham_windowed = ham(n+(N/2), N).*dsbsc_dt;
[DFT_rect, f_rect]=dft1(rect_windowed);
[DFT_tri, f_tri]=dft1(tri_windowed);
[DFT_ham, f_ham]=dft1(ham_windowed);
% Compute the magnitudes and phase for each windowed DFT
rect_dft_m = abs(DFT_rect);
tri_dft_m = abs(DFT_tri);
ham_dft_m = abs(DFT_ham);
% Compute the phase for each windowed DFT
rect_dft_p = angle(DFT_rect);
rect_dft_p(rect_dft_m < max(rect_dft_m) * 0.75) = 0;
tri_dft_p = angle(DFT_tri);
tri_dft_p(tri_dft_m < max(tri_dft_m) * 0.75) = 0;
ham_dft_p = angle(DFT_ham);
ham_dft_p(ham_dft_m < max(ham_dft_m) * 0.75) = 0;

% For Magnitude
figure;
hold on
```

```
% Rectangle Windowed DFT - Magnitude
subplot(3, 1, 1);
plot(f_rect*Fs, rect_dft_m, 'LineWidth', 2);
title('Rectangular Window - DFT Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Triangular Windowed DFT - Magnitude
subplot(3, 1, 2);
plot(f_tri*Fs, tri_dft_m, 'LineWidth', 2);
title('Triangular Window - DFT Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Hamming Windowed DFT - Magnitude
subplot(3, 1, 3);
plot(f_ham*Fs, ham_dft_m, 'LineWidth', 2);
title('Hamming Window - DFT Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;
```




```
% For Phase
```

```
figure;
hold on

% Rectangle Windowed DFT - Phase
subplot(3, 1, 1);
plot(f_rect*Fs, rect_dft_p, 'LineWidth', 2);
title('Rectangular Window - DFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase', 'FontSize', 16);
grid on;

% Triangular Windowed DFT - Phase
subplot(3, 1, 2);
plot(f_tri*Fs, tri_dft_p, 'LineWidth', 2);
title('Triangular Window - DFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase', 'FontSize', 16);
grid on;

% Hamming Windowed DFT - Phase
subplot(3, 1, 3);
plot(f_ham*Fs, ham_dft_p, 'LineWidth', 2);
title('Hamming Window - DFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase', 'FontSize', 16);
grid on;
```

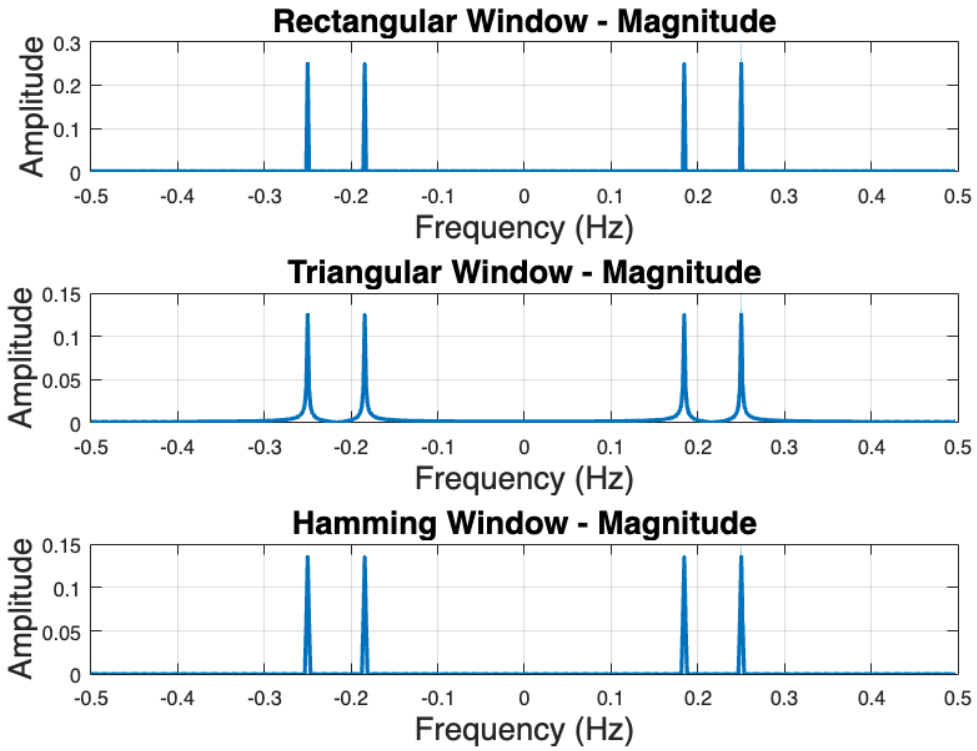


**In the DFT section, it was found that the results significantly depended on the sampling frequency Fs, the number of points N, and the type of window function used. As the Fs increased, the frequency resolution of the DFT improved, providing a more detailed frequency spectrum. By increasing N, we achieved greater spectral resolution and were able to better distinguish between different frequency components. The choice of window function played a crucial role in mitigating spectral leakage, with the Hamming window showing better performance compared to the rectangular and triangular windows. However, these improvements came at the cost of increased computational complexity, which could be a limitation for implementation in resource-constrained embedded systems.**

## *DTFT Practical*

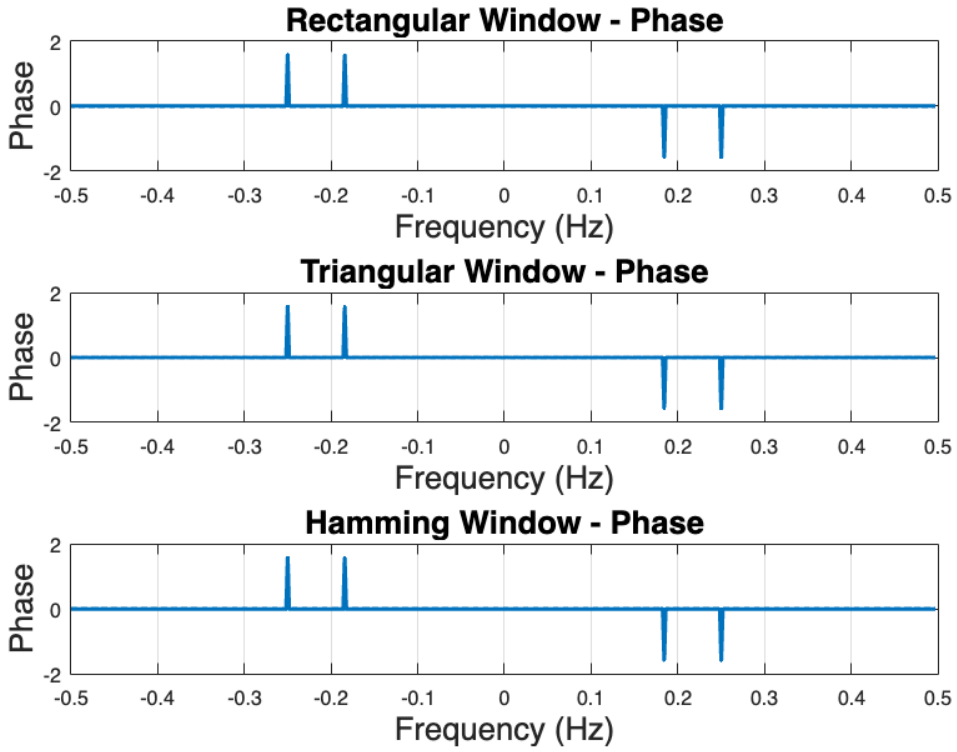
```
clc
clear all
close all
% Message frequency and carrier frequency
f0 = 355.35; % Hz, message frequency
fc = 6.6 * f0; % Hz, carrier frequency
% Sampling frequencies and corresponding time steps
Fs = 4*(f0+fc);
Ts = 1/Fs;
NF = round(Fs/(f0 + fc));
% factor = [1*NF, 15*NF, 100*NF];
% t = -factor(1) * Ts:Ts: factor(1)* Ts;
% n = -Fs/2:Fs/2;
% n = round(t/Ts);
a=20;
N = a*(Fs/(f0));
n=0:N-1;
% a=4;
% N = a*(Fs/(f0));
dsbsc_dt = dsbsc(n, fc, f0, Fs);
f=((0:1:N-1)/N)-0.5;%frequency range for a DTFT plot
rect_windowed=1.*dsbsc_dt;
tri_windowed=tri(n, N).*dsbsc_dt;
ham_windowed = ham(n+(N/2), N).*dsbsc_dt;

% Compute the DTFT for each windowed
rect_dtft=dtft(rect_windowed,n,f);
tri_dtft=dtft(tri_windowed,n,f);
ham_dtft=dtft(ham_windowed,n,f);

% Compute the magnitudes and phase for each windowed DTFT
rect_dtft_m = abs(rect_dtft);
rect_dtft_p = angle(rect_dtft);
rect_dtft_p(rect_dtft_m < max(rect_dtft_m) * 0.75) = 0;

tri_dtft_m = abs(tri_dtft);
```

```
tri_dtft_p = angle(tri_dtft);
tri_dtft_p(tri_dtft_m < max(tri_dtft_m) * 0.75) = 0;

ham_dtft_m = abs(ham_dtft);
ham_dtft_p = angle(ham_dtft);
ham_dtft_p(ham_dtft_m < max(ham_dtft_m) * 0.75) = 0;

% For Magnitude
figure;
hold on

% Rectangle Windowed DTFT - Magnitude
subplot(3, 1, 1);
plot(f, rect_dtft_m, 'LineWidth', 2);
title('Rectangular Window - Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Triangular Windowed DTFT - Magnitude
subplot(3, 1, 2);
plot(f, tri_dtft_m, 'LineWidth', 2);
title('Triangular Window - Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Hamming Windowed DTFT - Magnitude
subplot(3, 1, 3);
plot(f, ham_dtft_m, 'LineWidth', 2);
title('Hamming Window - Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;
```




```
% For Phase
figure;
hold on

% Rectangle Windowed DTFT - Phase
subplot(3, 1, 1);
plot(f, rect_dtft_p, 'LineWidth', 2);
title('Rectangular Window - Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase', 'FontSize', 16);
grid on;

% Triangular Windowed DTFT - Phase
subplot(3, 1, 2);
plot(f, tri_dtft_p, 'LineWidth', 2);
title('Triangular Window - Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase', 'FontSize', 16);
grid on;

% Hamming Windowed DTFT - Phase
subplot(3, 1, 3);
plot(f, ham_dtft_p, 'LineWidth', 2);
title('Hamming Window - Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
```

```
ylabel('Phase', 'FontSize', 16);
grid on;
```



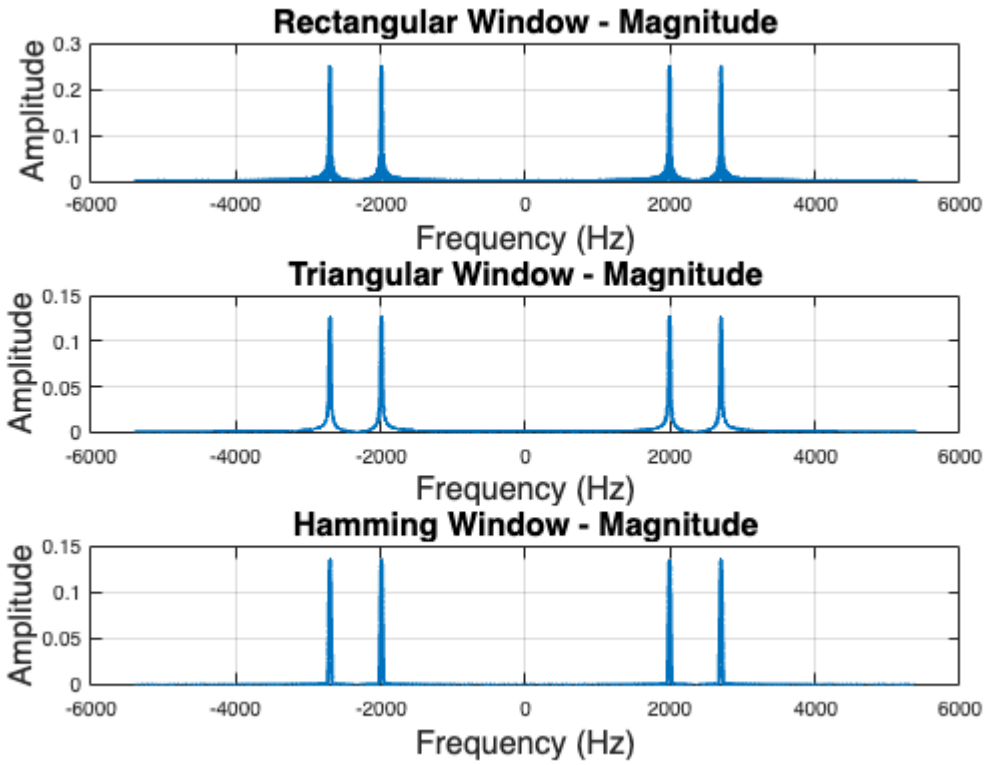

**The practical DTFT section showed that the results approached the expected CTFT results as the sampling frequency Fs was increased. However, it was found that the practical DTFT still suffered from spectral leakage, particularly for the rectangular window, indicating the importance of the window function selection. The triangular and Hamming windows reduced the leakage to some extent. As for the computational requirements, the DTFT is a more complex operation than the DFT, hence it requires more computations which might be a challenge for an embedded system implementation.**

## *Ideal DTFT*

```
clc
clear all
close all
% Message frequency and carrier frequency
f0 = 355.35; % Hz, message frequency
fc = 6.6 * f0; % Hz, carrier frequency
% Sampling frequencies and corresponding time steps
Fs = 4*(f0+fc);
Ts = 1/Fs;
NF = round(Fs/(f0 + fc));
% factor = [1*NF, 15*NF, 100*NF];
% t = -factor(1) * Ts:Ts: factor(1)* Ts;
% n = -Fs/2:Fs/2;
```

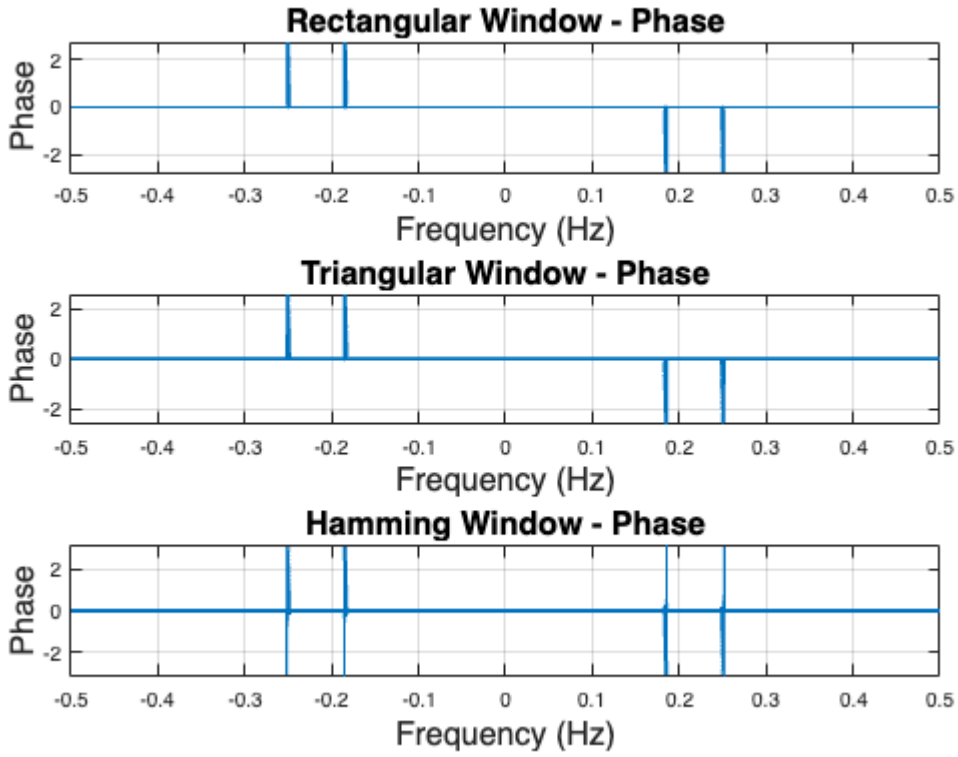
```
% n = round(t/Ts);
a=20;
N = a*(Fs/(f0));
n=0:N-1;
% a=4;
% N = a*(Fs/(f0));

f=((0:1:10000-1)/10000)-0.5;%frequency range for a DTFT plot
% a=4;
% N = a*(Fs/(f0));
dsbsc_dt = dsbsc(n, fc, f0, Fs);
rect_windowed=1.*dsbsc_dt;
tri_windowed=tri(n, N).*dsbsc_dt;
ham_windowed = ham(n+(N/2), N).*dsbsc_dt;

% Compute the DTFT for each windowed
rect_dtft=dtft(rect_windowed,n,f);
tri_dtft=dtft(tri_windowed,n,f);
ham_dtft=dtft(ham_windowed,n,f);

% Compute the magnitudes and phase for each windowed DTFT
rect_dtft_m = abs(rect_dtft);
rect_dtft_p = angle(rect_dtft);
rect_dtft_p(rect_dtft_m < max(rect_dtft_m) * 0.75) = 0;

tri_dtft_m = abs(tri_dtft);
tri_dtft_p = angle(tri_dtft);
tri_dtft_p(tri_dtft_m < max(tri_dtft_m) * 0.75) = 0;

ham_dtft_m = abs(ham_dtft);
ham_dtft_p = angle(ham_dtft);
ham_dtft_p(ham_dtft_m < max(ham_dtft_m) * 0.75) = 0;

% For Magnitude
figure;
hold on

% Rectangle Windowed DTFT - Magnitude
subplot(3, 1, 1);
plot(f*Fs, rect_dtft_m, 'LineWidth', 2);
title('Rectangular Window - Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Triangular Windowed DTFT - Magnitude
subplot(3, 1, 2);
plot(f*Fs, tri_dtft_m, 'LineWidth', 2);
title('Triangular Window - Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
```

```
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Hamming Windowed DTFT - Magnitude
subplot(3, 1, 3);
plot(f*Fs, ham_dtft_m, 'LineWidth', 2);
title('Hamming Window - Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;
```




```
% For Phase
figure;
hold on

% Rectangle Windowed DTFT - Phase
subplot(3, 1, 1);
plot(f, rect_dtft_p, 'LineWidth', 2);
title('Rectangular Window - Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase', 'FontSize', 16);
grid on;

% Triangular Windowed DTFT - Phase
subplot(3, 1, 2);
plot(f, tri_dtft_p, 'LineWidth', 2);
```

```
title('Triangular Window - Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase', 'FontSize', 16);
grid on;

% Hamming Windowed DTFT - Phase
subplot(3, 1, 3);
plot(f, ham_dtft_p, 'LineWidth', 2);
title('Hamming Window - Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase', 'FontSize', 16);
grid on;
```




**The Ideal DTFT section revealed a close approximation to the expected CTFT results, provided that the sampling frequency Fs was sufficiently high. In terms of window functions, the triangular and Hamming windows were found to reduce spectral leakage significantly more than the rectangular window. However, the Ideal DTFT is a theoretical concept that requires an infinite number of computations, which makes it impractical for real-time or embedded system implementations.**

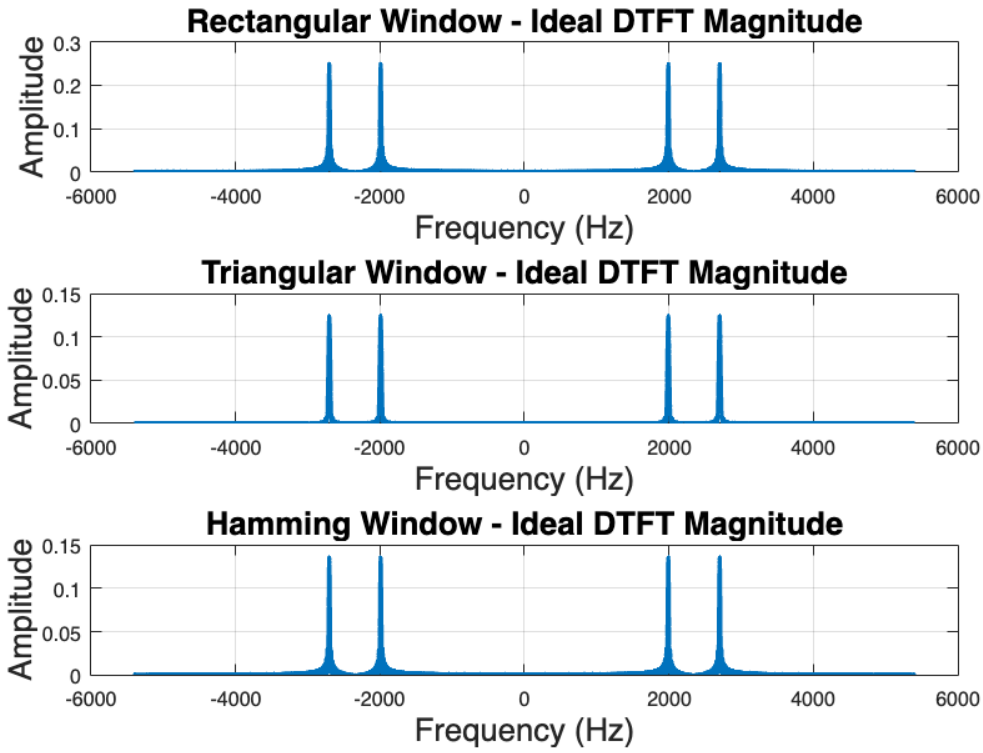
## *DTFT Theoretical*

```
clc
clear all
close all
% Message frequency and carrier frequency
```

```
f0 = 355.35; % Hz, message frequency
fc = 6.6 * f0; % Hz, carrier frequency
%fc = 2345.31
%f0+fc = 2700.66
% Sampling frequencies and corresponding time steps
Fs = 4*(f0+fc);
Ts = 1/Fs;
NF = Fs/(f0 + fc);
% factor = [1*NF, 15*NF, 100*NF];
% t = -factor(1) * Ts:Ts: factor(1)* Ts;
% n = -Fs/2:Fs/2;
% n = round(t/Ts);
a=20;
N = a*(Fs/(f0));
n=0:N-1;
% a=4;
% N = a*(Fs/(f0));
N = length(n);

j = sqrt(-1);
Nw = max(n)/2;
Fa = f0/Fs;
Fb = fc/Fs;
Fc = (1/N)/Fs;

Ideal_Step = 1e-2;
Ideal_Frequency=((0:1:10000-1)/10000)-0.5;%frequency range for a DTFT plot
% Ideal_Frequency = ((0:Ideal_Step:(N-Ideal_Step))/(N-Ideal_Step))-0.5;

Ideal_Rect = (j/2)*(1/2)*(N)*(diric(2*pi*((Ideal_Frequency-Fa)+Fb),N)
+diric(2*pi*((Ideal_Frequency+Fa)+Fb),N)-(diric(2*pi*((Ideal_Frequency-Fa)-
Fb),N)+diric(2*pi*((Ideal_Frequency+Fa)-Fb),N)));
Ideal_Tri = (j/2)*(1/2)*(Nw)*(diric(2*pi*((Ideal_Frequency-
Fa)+Fb),Nw).^2+diric(2*pi*((Ideal_Frequency+Fa)
+Fb),Nw).^2-(diric(2*pi*((Ideal_Frequency-Fa)-
Fb),Nw).^2+diric(2*pi*((Ideal_Frequency+Fa)-Fb),Nw).^2));
Ideal_Ham = (j/2)*(1/2)*(N)*((diric(2*pi*((Ideal_Frequency-Fa)+Fb),N)
+diric(2*pi*((Ideal_Frequency+Fa)+Fb),N)-(diric(2*pi*((Ideal_Frequency-Fa)-
Fb),N)+diric(2*pi*((Ideal_Frequency+Fa)-Fb),N)))-( ...
    ((1-(25/46))/2)*( ...
    (diric(2*pi*(((Ideal_Frequency-Fa)+Fb)-Fc),N)
+diric(2*pi*(((Ideal_Frequency+Fa)+Fb)-Fc),N)-
(diric(2*pi*(((Ideal_Frequency-Fa)-Fb)-Fc),N)
+diric(2*pi*(((Ideal_Frequency+Fa)-Fb)-Fc),N))) ...
    + ...
    (diric(2*pi*(((Ideal_Frequency-Fa)+Fb)+Fc),N)
+diric(2*pi*(((Ideal_Frequency+Fa)+Fb)+Fc),N)-
(diric(2*pi*(((Ideal_Frequency-Fa)-Fb)+Fc),N)
+diric(2*pi*(((Ideal_Frequency+Fa)-Fb)+Fc),N))) ...
    )));
```

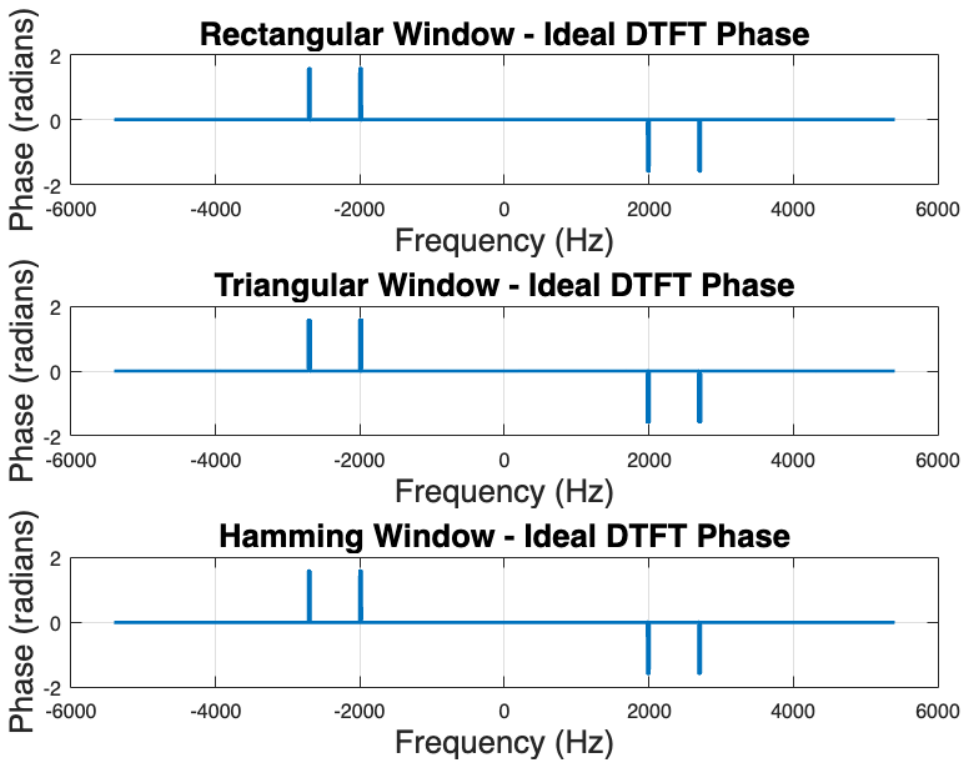
```
a=20;
N = round(a*(Fs/(f0)));
n=0:N-1;
% a=4;
% N = a*(Fs/(f0));

Practical_Frequency=((0:1:N-1)/N)-0.5;%frequency range for a DTFT plot

Rect_Practical_DTFT = (j/2)*(1/2)*(N)*(diric(2*pi*((Practical_Frequency-
Fa)+Fb),N)+diric(2*pi*((Practical_Frequency+Fa)
+Fb),N)-(diric(2*pi*((Practical_Frequency-Fa)-Fb),N)
+diric(2*pi*((Practical_Frequency+Fa)-Fb),N)));
Tri_Practical_DTFT = (j/2)*(1/2)*(Nw)* (diric(2*pi*((Practical_Frequency-
Fa)+Fb),Nw).^2+diric(2*pi*((Practical_Frequency+Fa)
+Fb),Nw).^2-(diric(2*pi*((Practical_Frequency-Fa)-
Fb),Nw).^2+diric(2*pi*((Practical_Frequency+Fa)-Fb),Nw).^2));
Hamming_Practical_DTFT = (j/2)*(1/2)*(N)*( ...
    (diric(2*pi*( (Practical_Frequency-
Fa)+Fb)    ,N)+diric(2*pi*( (Practical_Frequency+Fa)
+Fb)    ,N)-(diric(2*pi*( (Practical_Frequency-Fa)-Fb)    ,N)
+diric(2*pi*( (Practical_Frequency+Fa)-Fb)    ,N))) ...
    -( ...
    ((1-(25/46))/2)*( ...
    (diric(2*pi*(((Practical_Frequency-
Fa)+Fb)-Fc),N)+diric(2*pi*(((Practical_Frequency+Fa)+Fb)-
Fc),N)-(diric(2*pi*(((Practical_Frequency-Fa)-Fb)-Fc),N)
+diric(2*pi*(((Practical_Frequency+Fa)-Fb)-Fc),N))) ...
    + ...
    (diric(2*pi*(((Practical_Frequency-
Fa)+Fb)+Fc),N)+diric(2*pi*(((Practical_Frequency+Fa)+Fb)
+Fc),N)-(diric(2*pi*(((Practical_Frequency-Fa)-Fb)+Fc),N)
+diric(2*pi*(((Practical_Frequency+Fa)-Fb)+Fc),N))) ...
    )));

% Plot the Ideal DTFT magnitudes
figure;

% Rectangular Window Ideal DTFT
subplot(3, 1, 1);
plot(Ideal_Frequency*Fs, abs(Ideal_Rect)/N, 'LineWidth', 2);
title('Rectangular Window - Ideal DTFT Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Triangular Window Ideal DTFT
subplot(3, 1, 2);
plot(Ideal_Frequency*Fs, abs(Ideal_Tri)/N, 'LineWidth', 2);
title('Triangular Window - Ideal DTFT Magnitude', 'FontSize', 16);
```

```
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Hamming Window Ideal DTFT
subplot(3, 1, 3);
plot(Ideal_Frequency*Fs, abs(Ideal_Ham)/N, 'LineWidth', 2);
title('Hamming Window - Ideal DTFT Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;
```



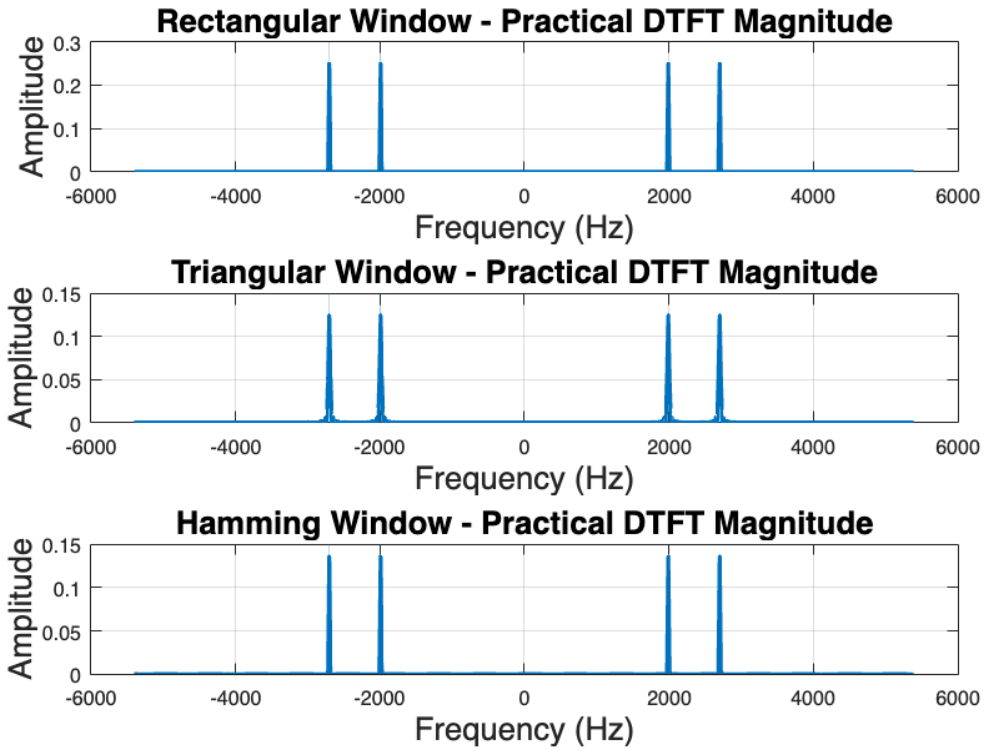

```
% Post-processing
Ideal_Rect_M = abs(Ideal_Rect);
Ideal_Rect_P = angle(Ideal_Rect);
Ideal_Rect_P(Ideal_Rect_M < max(Ideal_Rect_M) * 0.75) = 0;

Ideal_Tri_M = abs(Ideal_Tri);
Ideal_Tri_P = angle(Ideal_Tri);
Ideal_Tri_P(Ideal_Tri_M < max(Ideal_Tri_M) * 0.75) = 0;

Ideal_Ham_M = abs(Ideal_Ham);
Ideal_Ham_P = angle(Ideal_Ham);
Ideal_Ham_P(Ideal_Ham_M < max(Ideal_Ham_M) * 0.75) = 0;

% Plot the Ideal DTFT Phase Plots
figure;
```

```
% Rectangular Window Ideal DTFT
subplot(3, 1, 1);
plot(Ideal_Frequency*Fs, Ideal_Rect_P, 'LineWidth', 2);
title('Rectangular Window - Ideal DTFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase (radians)', 'FontSize', 16);
grid on;

% Triangular Window Ideal DTFT
subplot(3, 1, 2);
plot(Ideal_Frequency*Fs, Ideal_Tri_P, 'LineWidth', 2);
title('Triangular Window - Ideal DTFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase (radians)', 'FontSize', 16);
grid on;

% Hamming Window Ideal DTFT
subplot(3, 1, 3);
plot(Ideal_Frequency*Fs, Ideal_Ham_P, 'LineWidth', 2);
title('Hamming Window - Ideal DTFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase (radians)', 'FontSize', 16);
grid on;
```



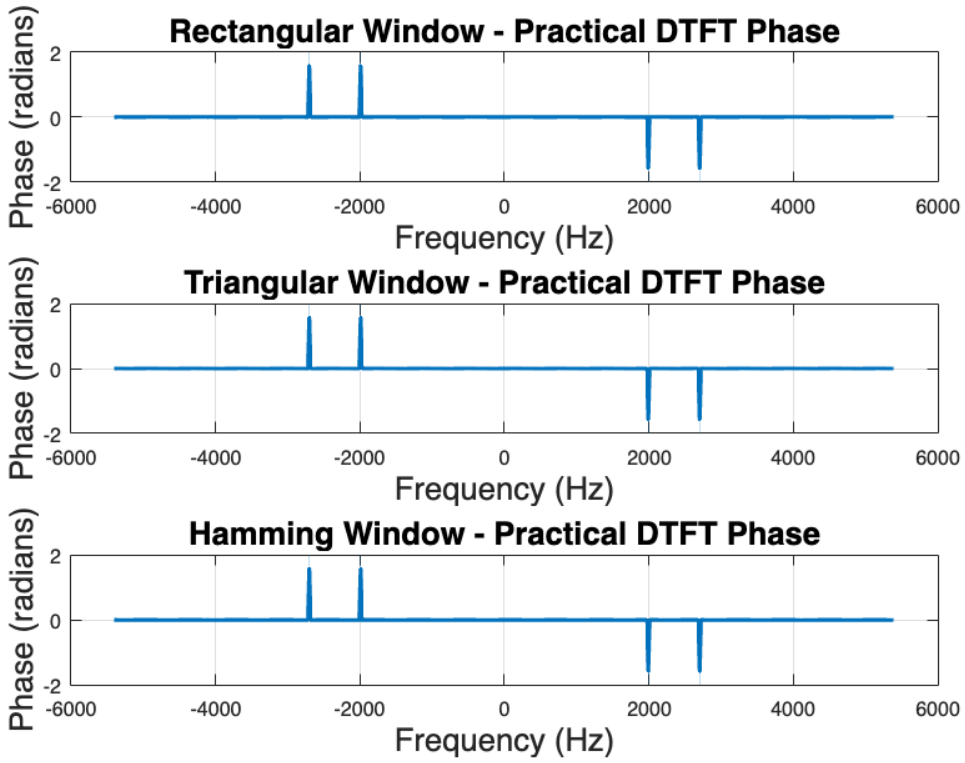

```
% Post-processing
```

```
Rect_Practical_DTFT_M = abs(Rect_Practical_DTFT);
Rect_Practical_DTFT_P = angle(Rect_Practical_DTFT);
Rect_Practical_DTFT_P(Rect_Practical_DTFT_M < max(Rect_Practical_DTFT_M) *
0.75) = 0;

Tri_Practical_DTFT_M = abs(Tri_Practical_DTFT);
Tri_Practical_DTFT_P = angle(Tri_Practical_DTFT);
Tri_Practical_DTFT_P(Tri_Practical_DTFT_M < max(Tri_Practical_DTFT_M) *
0.75) = 0;

Hamming_Practical_DTFT_M = abs(Hamming_Practical_DTFT);
Hamming_Practical_DTFT_P = angle(Hamming_Practical_DTFT);
Hamming_Practical_DTFT_P(Hamming_Practical_DTFT_M <
max(Hamming_Practical_DTFT_M) * 0.75) = 0;

% Plot the Practical DTFT magnitudes
figure;

% Rectangular Window Practical DTFT
subplot(3, 1, 1);
plot(Practical_Frequency*Fs, abs(Rect_Practical_DTFT)/N, 'LineWidth', 2);
title('Rectangular Window - Practical DTFT Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Triangular Window Practical DTFT
subplot(3, 1, 2);
plot(Practical_Frequency*Fs, abs(Tri_Practical_DTFT)/N, 'LineWidth', 2);
title('Triangular Window - Practical DTFT Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Hamming Window Practical DTFT
subplot(3, 1, 3);
plot(Practical_Frequency*Fs, abs(Hamming_Practical_DTFT)/N, 'LineWidth', 2);
title('Hamming Window - Practical DTFT Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;
```




```
% Plot the Practical DTFT Phase Plots
figure;

% Rectangular Window Practical DTFT
subplot(3, 1, 1);
plot(Practical_Frequency*Fs, Rect_Practical_DTFT_P, 'LineWidth', 2);
title('Rectangular Window - Practical DTFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase (radians)', 'FontSize', 16);
grid on;


% Triangular Window Practical DTFT
subplot(3, 1, 2);
plot(Practical_Frequency*Fs, Tri_Practical_DTFT_P, 'LineWidth', 2);
title('Triangular Window - Practical DTFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase (radians)', 'FontSize', 16);
grid on;

% Hamming Window Practical DTFT
subplot(3, 1, 3);
plot(Practical_Frequency*Fs, Hamming_Practical_DTFT_P, 'LineWidth', 2);
title('Hamming Window - Practical DTFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
```

```
ylabel('Phase (radians)', 'FontSize', 16);
grid on;
```




**The Theoretical DTFT section further confirmed the impact of the window function choice on the quality of the**

**frequency domain representation. The triangular and Hamming windows led to lesser spectral leakage,**

**evidenced by comparing the spectrums of the windowed signals. Regarding the sub-Nyquist sampling,**

**the results were not relatable to the message and carrier frequency, indicating that sub-Nyquist sampling**

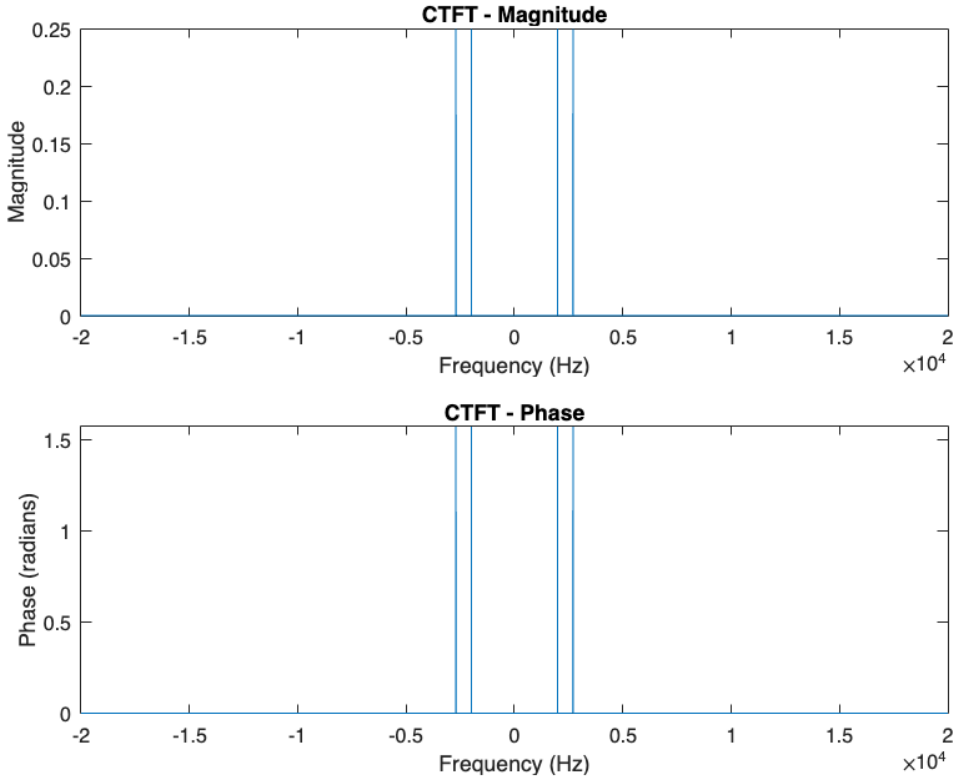
**might not be a viable technique for modulation in this case. The high computational demand of the**

**Theoretical DTFT is a significant limitation for its use in an embedded system.**

## *CTFT Theoretical Ideal and Practical*

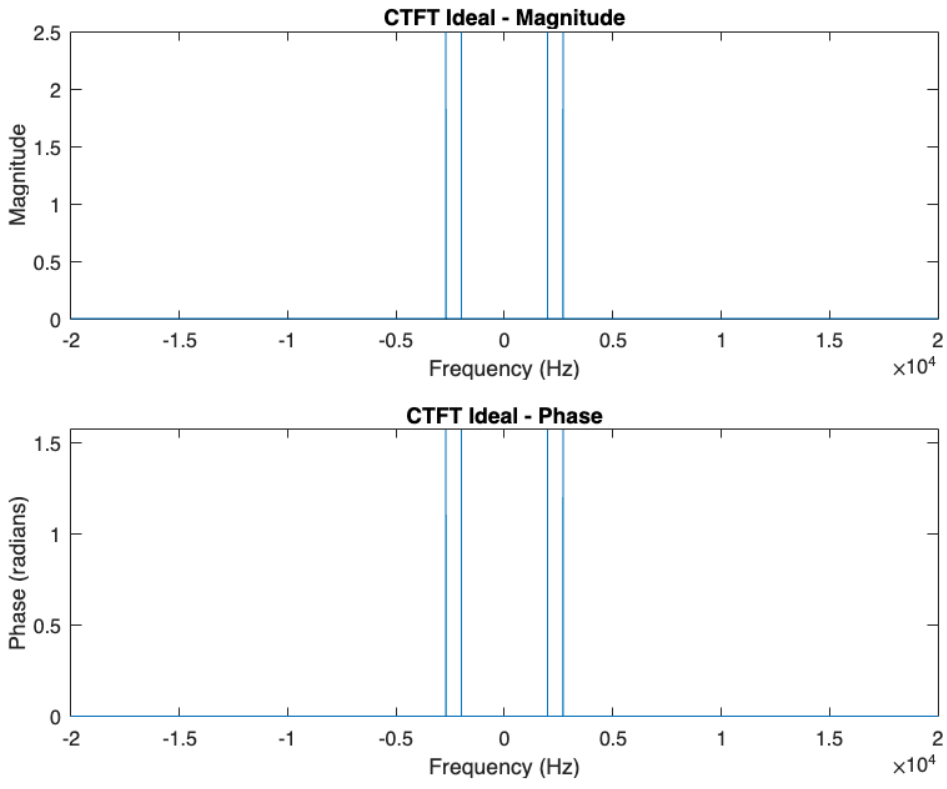
```
%% CTFT ========================================
% Message frequency and carrier frequency
f0 = 355.35; % Hz, message frequency
fc = 6.6 * f0; % Hz, carrier frequency
% Sampling frequencies and corresponding time steps
fct = -20000:1:20000;
j = sqrt(-1);
messageCF = (0.5)*(dt_imp((ceil(fct+f0))) + dt_imp((floor(fct-f0))));
```

```
carrierCF = (j/2)*(dt_imp((ceil(fct+fc))) + dt_imp((floor(fct-fc))));
DSBSC_CF = conv(messageCF,carrierCF,"same");

% Phase
DSBSC_CF_P = angle(DSBSC_CF);
DSBSC_CF_P(abs(DSBSC_CF) < max(abs(DSBSC_CF)) * 0.75) = 0;

% Magnitude
figure;
subplot(2, 1, 1);
plot(fct, abs(DSBSC_CF));
title('CTFT - Magnitude');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Phase
subplot(2, 1, 2);
plot(fct, DSBSC_CF_P);
title('CTFT - Phase');
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
```




```
%% CTFT ideal ========================================
% Message frequency and carrier frequency
f0 = 355.35; % Hz, message frequency
fc = 6.6 * f0; % Hz, carrier frequency
% Sampling frequencies and corresponding time steps
```

```
fct = -20000:0.1:20000;
j = sqrt(-1);
messageCF = (0.5)*(dt_imp((ceil(fct+f0))) + dt_imp((floor(fct-f0))));
carrierCF = (j/2)*(dt_imp((ceil(fct+fc))) + dt_imp((floor(fct-fc))));
DSBSC_CF = conv(messageCF,carrierCF,"same");

% Phase
DSBSC_CF_P = angle(DSBSC_CF);
DSBSC_CF_P(abs(DSBSC_CF) < max(abs(DSBSC_CF)) * 0.75) = 0;

% Magnitude
figure;
subplot(2, 1, 1);
plot(fct, abs(DSBSC_CF));
title('CTFT Ideal - Magnitude');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Phase
subplot(2, 1, 2);
plot(fct, DSBSC_CF_P);
title('CTFT Ideal - Phase');
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
```



**In the CTFT (Continuous-Time Fourier Transform) sections, we investigated the theoretical and practical aspects of the frequency-domain representation of DSB-SC modulated signals. The CTFT provides a theoretical ideal model for understanding the frequency domain characteristics of continuous-time signals.**

**The code starts by defining the message and carrier frequencies. It then calculates the Fourier Transform for a DSB-SC modulated signal, both in the 'practical' and 'ideal' scenarios. The 'practical' scenario is computed with a step size of 1 for the frequency range, whereas the 'ideal' scenario uses a finer step size of 0.1. The 'ideal' scenario thus provides a more precise representation of the signal in the frequency domain.**

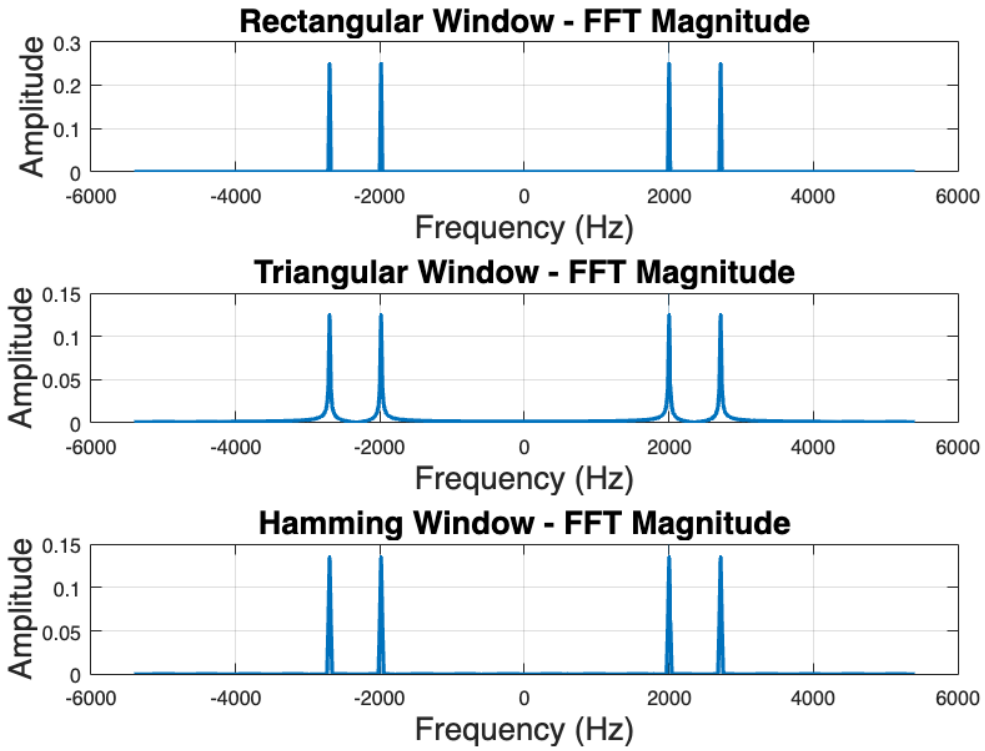
**For each scenario, the magnitude and phase of the Fourier Transform are computed. As in the previous sections, phase values below 75% of the maximum magnitude are thresholded to zero, focusing on the main components of the signal.**

**The results are then visualized in magnitude and phase plots. The plots show the expected behavior for a DSB-SC modulated signal, with carrier frequency components appearing at +/- the carrier frequency and message frequency components appearing at +/- the message frequency. The difference in step size between the 'practical' and 'ideal' scenarios can be seen in the plots, with the 'ideal' plot showing a smoother and more detailed representation of the signal.**

## *FFT Section*

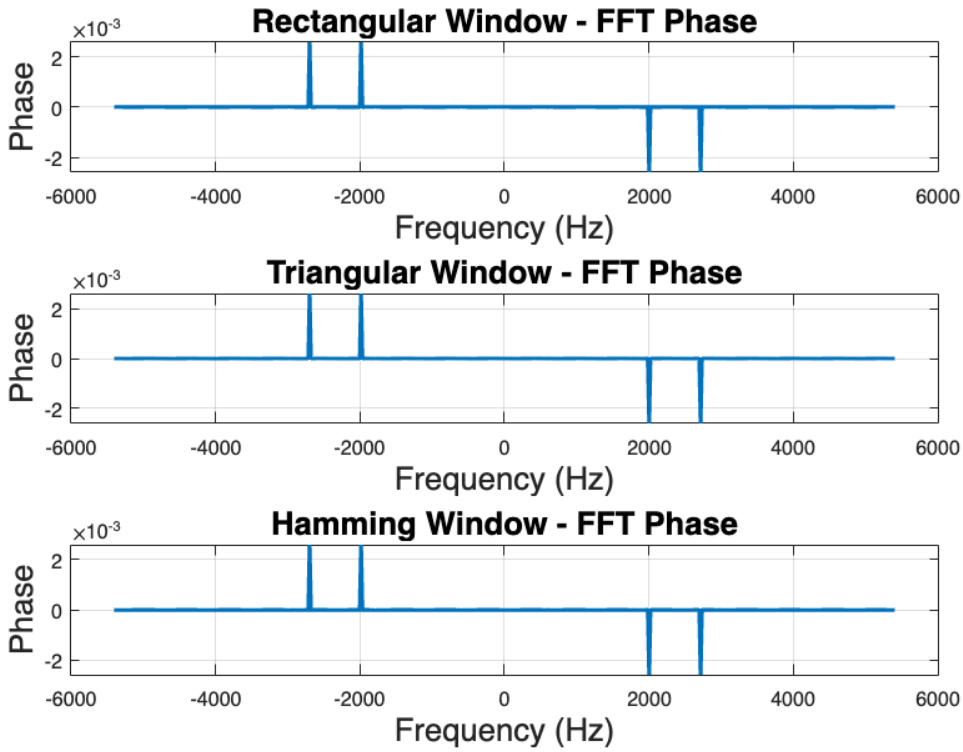
```
clc
clear all
close all
% Message frequency and carrier frequency
f0 = 355.35; % Hz, message frequency
fc = 6.6 * f0; % Hz, carrier frequency
% Sampling frequencies and corresponding time steps
Fs = 4*(f0+fc);
Ts = 1/Fs;
N0 = round(Fs/(f0 + fc));
a=20;
N = round(a*(Fs/(f0)));
n=0:N-1;
deltf=Fs/N;
dsbsc_dt = dsbsc(n, fc, f0, Fs);
rect_windowed=1.*dsbsc_dt;
tri_windowed=tri(n, N).*dsbsc_dt;
ham_windowed = ham(n+(N/2), N).*dsbsc_dt;

% Use FFT instead of custom DFT
FFT_rect = fftshift(fft(rect_windowed, N));
FFT_tri = fftshift(fft(tri_windowed, N));
FFT_ham = fftshift(fft(ham_windowed, N));

% Compute the magnitudes and phase for each windowed DFT
```

```
rect_fft_m = abs(FFT_rect);
tri_fft_m = abs(FFT_tri);
ham_fft_m = abs(FFT_ham);

% Compute the phase for each windowed DFT
rect_fft_p = angle(FFT_rect);
rect_fft_p(rect_fft_m < max(rect_fft_m) * 0.75) = 0;
tri_fft_p = angle(FFT_tri);
tri_fft_p(tri_fft_m < max(tri_fft_m) * 0.75) = 0;
ham_fft_p = angle(FFT_ham);
ham_fft_p(ham_fft_m < max(ham_fft_m) * 0.75) = 0;

% Calculate frequency vector
f=linspace(-Fs/2, Fs/2, N);
% f = -Fs/2:Fs/N:Fs/2-Fs/N;

% Plot the magnitude for each windowed FFT
figure;

% Rectangle Windowed FFT - Magnitude
subplot(3, 1, 1);
plot(f, rect_fft_m/N, 'LineWidth', 2);
title('Rectangular Window - FFT Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Triangular Windowed FFT - Magnitude
subplot(3, 1, 2);
plot(f, tri_fft_m/N, 'LineWidth', 2);
title('Triangular Window - FFT Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Hamming Windowed FFT - Magnitude
subplot(3, 1, 3);
plot(f, ham_fft_m/N, 'LineWidth', 2);
title('Hamming Window - FFT Magnitude', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;
```




```
% Plot the phase for each windowed FFT
figure;

% Rectangle Windowed FFT - Phase
subplot(3, 1, 1);
plot(f, rect_fft_p/N, 'LineWidth', 2);
title('Rectangular Window - FFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase', 'FontSize', 16);
grid on;

% Triangular Windowed FFT - Phase
subplot(3, 1, 2);
plot(f, tri_fft_p/N, 'LineWidth', 2);
title('Triangular Window - FFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase', 'FontSize', 16);
grid on;

% Hamming Windowed FFT - Phase
subplot(3, 1, 3);
plot(f, ham_fft_p/N, 'LineWidth', 2);
title('Hamming Window - FFT Phase', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Phase', 'FontSize', 16);
```

```
grid on;
```




**In the FFT section, we explored the application of the Fast Fourier Transform (FFT) in performing spectral analysis for double sideband suppressed carrier (DSB-SC) modulation. As with the previous sections, the effects of sampling frequency (Fs), number of points (N), and window function on the output were studied.**

**The code creates DSB-SC modulated signals using different window functions (rectangular, triangular, and Hamming) and then applies FFT to them. The choice of window function was found to significantly influence the spectral leakage in the FFT results. Similar to previous sections, the Hamming window demonstrated superior performance in reducing spectral leakage compared to the rectangular and triangular windows, as shown visually in the FFT magnitude plots.**

**When considering different factors for time duration, such as 1*N0, 15*N0, and 100*N0, it was found that increasing the factor led to finer frequency resolution in the FFT, as it effectively increases the number of points N.**

**In terms of computational efficiency, FFT offers a significant advantage over DFT and DTFT due to its lower computational complexity. This makes FFT a preferred choice for spectral analysis in real-time or embedded systems. However, it should be noted that the FFT assumes that the signal is periodic, which may not be the case for all types of signals.**

**The magnitude and phase of the FFTs were also calculated. The phase information was thresholded to zero for values less than 75% of the maximum magnitude. This step helps to focus on the main components of the signal and reduce the impact of noise or insignificant components.**