4.2 Numerical Approximation to the Continuous-Time Fourier Transform

A large class of signals can be represented using the continuous-time Fourier transform (CTFT) in Eq. (4.1). In this exercise you will use MATLAB to compute numerical approximations to the CTFT integral, Eq. (4.2). By approximating the integral using a summation over closely spaced samples in **t**, you will be able to use the function **fft** to compute your approximation very efficiently. The approximation you will use follows from the definition of the integral

$$\int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt = \lim_{\tau\to\infty}\sum_{n=\infty}^{\infty} x(n\tau)e^{-j\omega n\tau}\tau$$

For a large class of signals and for sufficiently small τ, the sum on the right-hand side is a good approximation to the CTFT integral. If the signal $x(t)$ is equal to zero for $t<0$ and $t\geq T$, then the approximation can be written

(4.7)

$$\int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt = \int_{0}^{T} x(t)e^{-j\omega t}dt \approx \lim_{\tau\to\infty}\sum_{n=0}^{N-1} x(n\tau)e^{-j\omega n\tau}\tau$$

where $T=N\tau$ and N is an integer. You can use the function **fft** to compute the sum in Eq. (4.7) for a discrete set of frequencies ω_k. If the N samples $x(n\tau)$ are stored in the vector **x** then the function call **X=tau*fft(x)** calculates

(4.8)

$$X(k+1) = \tau\sum_{n=0}^{N-1} x(n\tau)e^{-j\omega n\tau}, \qquad 0\leq k\leq N$$

(4.9)

$$\approx X(j\omega_k)$$

where

4.10()

$$\omega_k = \begin{cases} \dfrac{2\pi k}{N\tau}, & 0\leq k\leq \dfrac{N}{2} \\ \dfrac{2\pi k}{N\tau}-\dfrac{2\pi}{\tau}, & \dfrac{N}{2}+1\leq k\leq N \end{cases}$$

and **N** is assumed to be even. For reasons of computational efficiency, **fft** returns the positive frequency samples before the negative frequency samples. To place the frequency samples in ascending order, you can

use the function **fftshift**. To store in **X** samples of $X(j\omega_k)$ ordered such that $x(k+1)$ is the CTFT evaluated at $-\pi/\tau + (2\pi k/N\tau)$ for $0 \le k \le N-1$, use ***X = fftshift (tau * fft(x))*.**

For this exercise, you will approximate the CTFT of $x(t) = e^{-2|t|}$ using the function **fft** and a truncated version of $x(t)$. You will see that for sufficiently small τ, you can compute an accurate numerical approximation to $X(j\omega)$.

Basic Problems

(a). Find an analytic expression for the CTFT of $x(t) = e^{-2|t|}$. You may find it helpful to think of $x(t) = g(t) + g(-t)$, where $g(t) = e^{2t}u(t)$.

$x(t) = e^{-2|t|}$

$x(t) = e^{-a|t|} = \{e^{-at}, t > 0 e^{at}; t < 0$

$$x(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}$$

$$= \int_{-\infty}^{\infty} e^{at}e^{-j\omega t}dt + \int_{0}^{\infty} e^{-at}e^{-j\omega t}dt$$

$$= \int_{-\infty}^{0} e^{(a-j\omega)t}dt + \int_{0}^{\infty} e^{-(a+j\omega)t}dt$$

$$x(\omega) = \frac{1}{a-j\omega} + \frac{1}{a+j\omega} = \frac{2a}{a^2+\omega^2}$$

a=2 : $x(\omega) = \frac{2x2}{2^2+\omega^2}$

$$x(\omega) = \frac{4}{4+\omega^2}$$

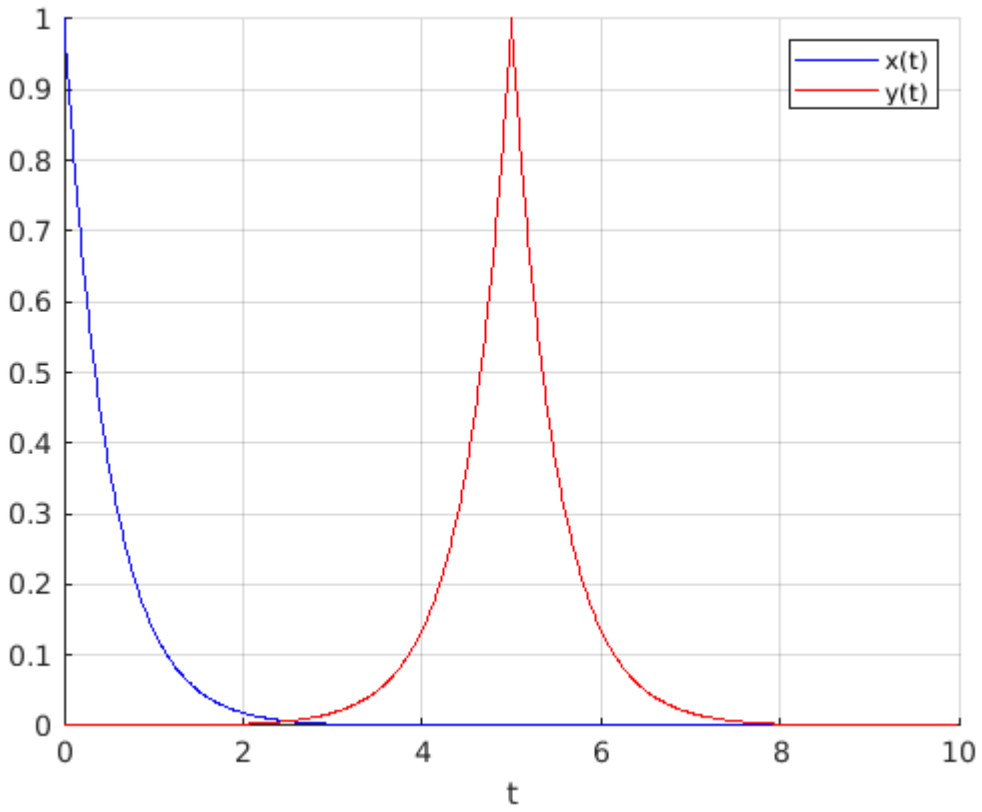
4.2a Analysis

(b). Create a vector containing samples of the signal $y(t) = x(t-5)$ for $\tau = 0.01$ and **T = 10** over the range **t=[O:tau:T-tau]**. Since $x(t)$ is effectively zero for $|t| > 5$, you can calculate the CTFT of the signal $y(t) = x(t-5)$ from the above analysis using $N = T/\tau$. Your vector **y** should have length N.

```
%4.2b Code
tau = 0.01;
T = 10;
t = 0:tau:T-tau;


%N= length(t);
%for k = 1:length(t)
%    y(k) = exp(-2*abs(t(k)-5));
%end
%figure;
%plot(t, y);

% ==========================================
% REVISION
x = exp(-2*abs(t));
y=exp(-2*abs(t-5));
figure;
hold;
```

```
Current plot held
```

```


plot(t, x, 'b');
xlabel('t');
grid;
plot(t, y, 'r');
legend('x(t)', 'y(t)');
hold off;
```




```
% ===========================================
```

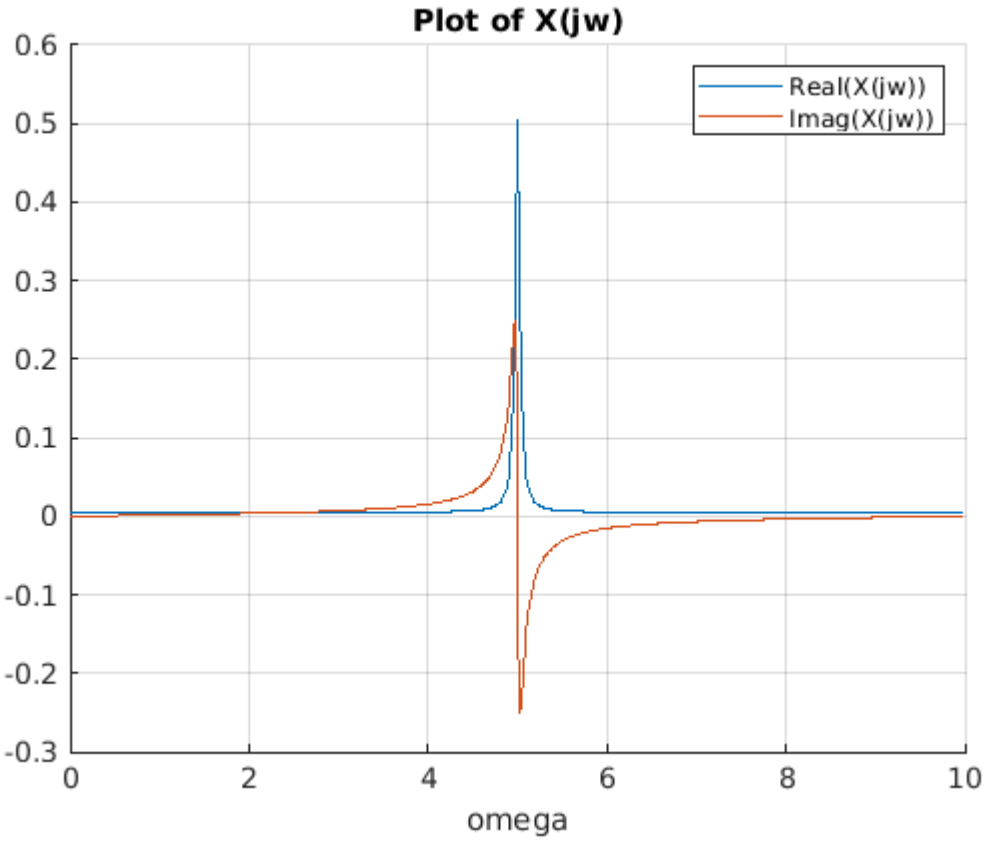
(c). Calculate samples $Y(jwk)$ by typing **Y=fftshift(tau*fft(y))**.

```
%4.2c Code

%Y = fftshift(tau*fft(y));

%===========================================
%REVISION
clf;
figure;
hold;
```

```
Current plot held
```

```
X=fftshift(tau*fft(x));
plot(t, real(X));
grid;
plot(t, imag(X));
title('Plot of X(jw)');
legend('Real(X(jw))', 'Imag(X(jw))');
```

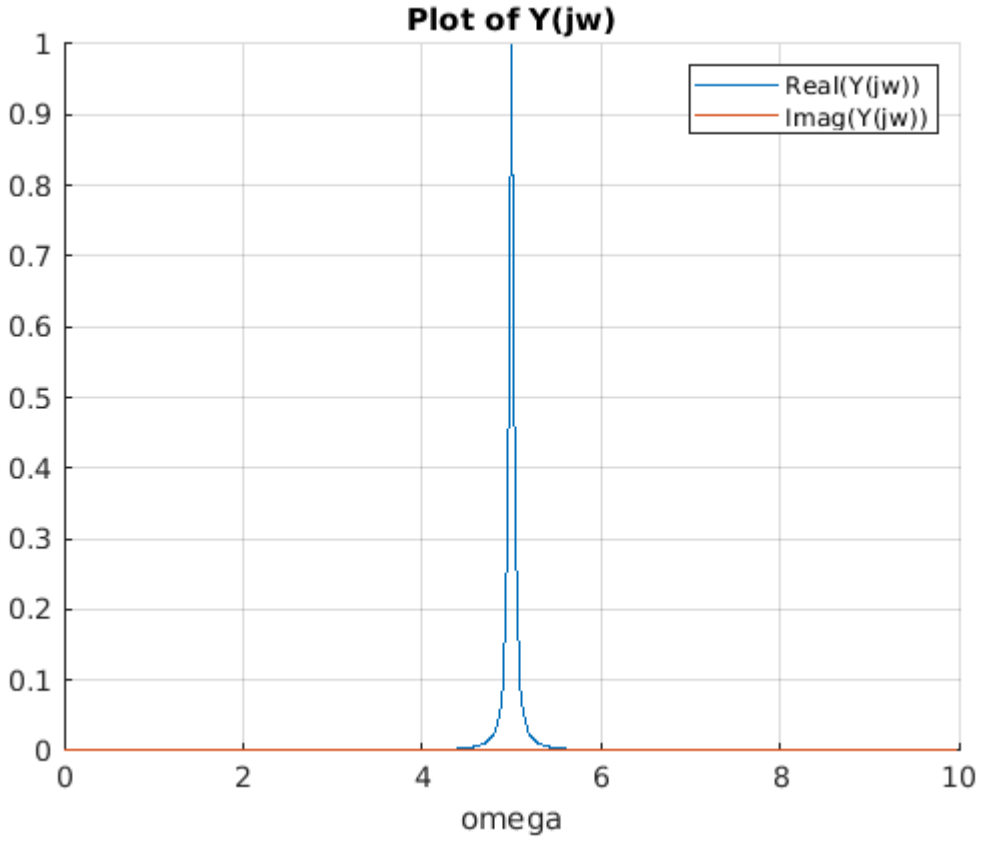
```
xlabel('omega');
```




```
figure;
hold;
```

```
Current plot held
```

```
Y=fftshift(tau*fft(y));
plot(t, abs(real(Y)));
grid;
plot(t, imag(Y));
title('Plot of Y(jw)');
legend('Real(Y(jw))', 'Imag(Y(jw))');
xlabel('omega');
```




```
%=========================================
```

(d). Construct a vector **w** of frequency samples that correspond to the values stored in the vector **Y** as follows

>> w = - (pi/tau) + (0:N-1) * (2*pi / (N*tau));

```
%4.2d Code
w = -(pi/tau) + (1:N)*(2*pi/(N*tau));
```

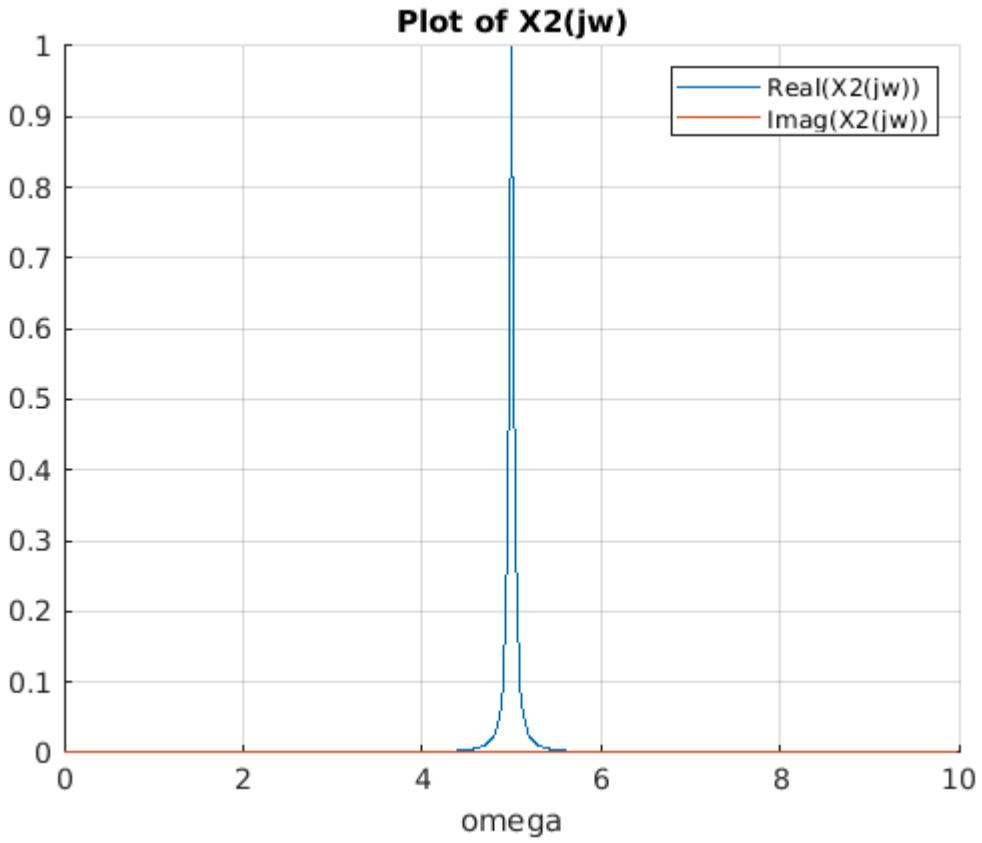
(e). Since $y(t)$ is related to $x(t)$ through a time shift, the CTFT $X(j\omega)$ is related to $Y(j\omega)$ by a linear phase term of the form $e^{j5\omega}$. Using the frequency vector **w**, compute samples of $X(j\omega)$ directly from **Y**, storing the result in the vector **X**.

```
%4.2e Code

%X = Y.*exp(1j*5*w);

%==================================
```

```
% REVISION
clf;
N = T/tau;
w = -(pi/tau) + (0:N-1)*(2*pi/(N*tau));
X2=Y.*exp(1j*5*w);

figure;
hold;
```

```
Current plot held
```

```
plot(t, real(X2));
grid;
plot(t, imag(X2));
title('Plot of X2(jw)');
legend('Real(X2(jw))', 'Imag(X2(jw))');
xlabel('omega');
```

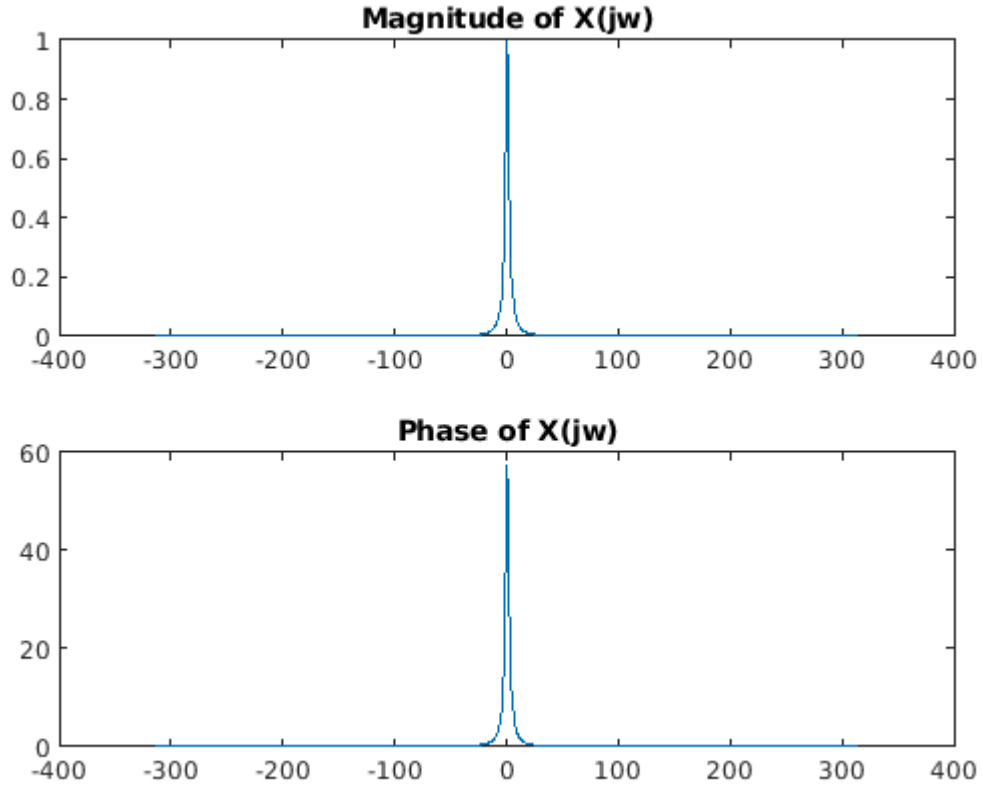



```
%===================================
```

(f). Using **abs** and **angle**, plot the magnitude and phase of **X** over the frequency range specified in **w**. For the same values of ω, also plot the magnitude and phase of the analytic expression you derived for $X(j\omega)$ in Part (a). Does your approximation of the CTFT match what you derived analytically? If you plot the magnitude on a

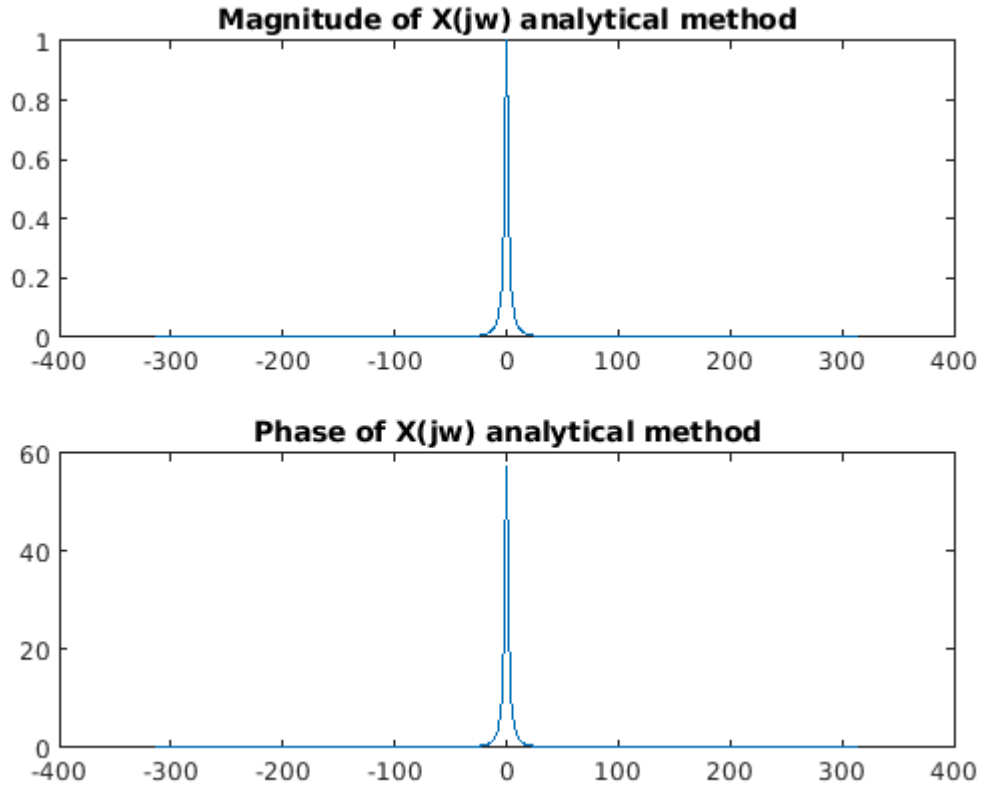
logarithmic scale, using semilogy, you will notice that at higher frequencies the approximation is not as good as at lower frequencies. Since you have approximated $x(t)$ with samples $x(n\tau)$, your approximation will be better for frequency components of the signal that do not vary much over time intervals of length τ.

```
%4.2f Code
clf;

figure;

subplot(2, 1, 1);
plot(w, abs(X));
title('Magnitude of X(jw)');

subplot(2, 1, 2);
plot(w, angle(X));
title('Phase of X(jw)');
```



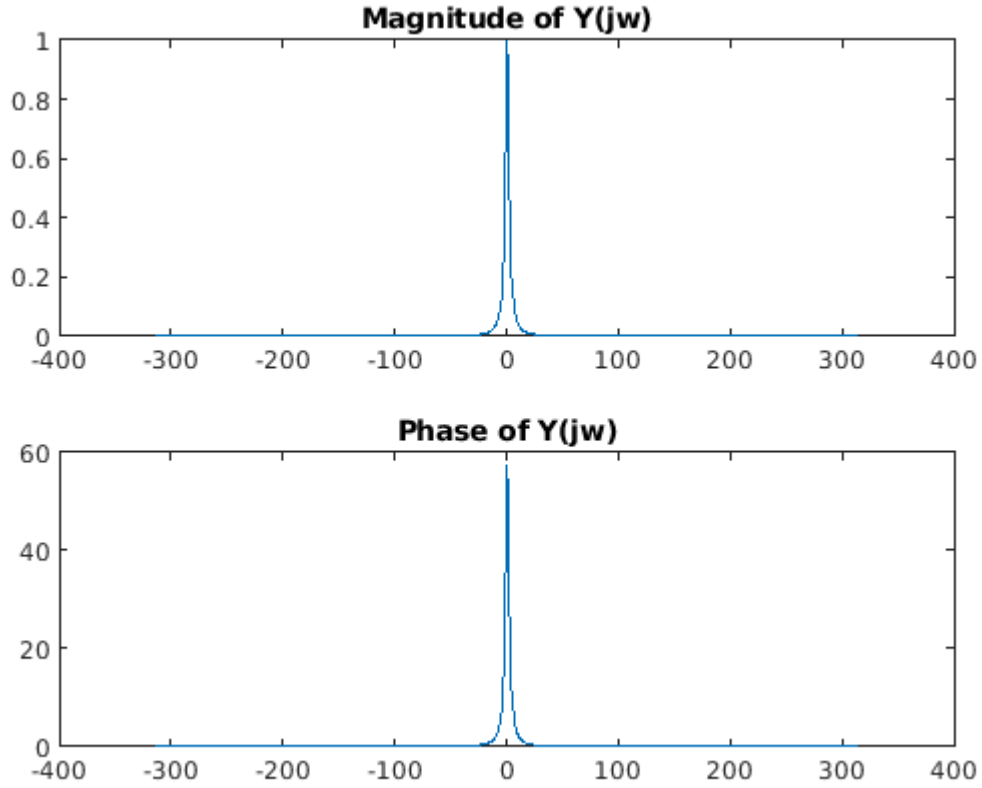

```
%figure;
%for k = 1:length(w)
%    X_analytical(k) = 4/(4+w(k)^2);
%end

subplot(2, 1, 1);
plot(w, X_analytical);
title('Magnitude of X(jw) analytical method');
```

```
subplot(2, 1, 2);
plot(w, X_analytical*57.3);
title('Phase of X(jw) analytical method');
```



(g). Plot the magnitude and phase of **Y** using **abs** and **angle**. How do they compare with **X**? Could you have anticipated this result?

```
%4.2g Code
figure;

subplot(2, 1, 1);
plot(w, abs(Y));
title('Magnitude of Y(jw)');

subplot(2, 1, 2);
plot(w, abs(Y*57.3));
title('Phase of Y(jw)');
```



4.3 Properties of the Continuous-Time Fourier Transform

In this exercise, you will enhance your understanding of the continuous-time Fourier transform (CTFT) by analyzing and manipulating audio signals in the frequency and time domains. Audio signals in MATLAB are represented by vectors containing samples of the continuous-time audio signal. The sampling rate is assumed to be 8192 Hz, i.e., the audio

signal is sampled every $\Delta t = (1/8192)$ seconds. To be more precise, for an audio signal $y(t)$ sampled at 8192 Hz over the interval $0 \leq t \leq N\Delta t$, the N-element vector **y** which represents the audio signal is given by

$$y(n) = y((n-1)\Delta t), \quad n = 1, ..., N.$$

The function **sound** can then be used to play the signal on your computer's speaker. While this is a sampled representation of a continuous-time audio signal $y(t)$, if $y(t)$ is zero otside the sampling interval and the sampling rate $f_s = 8192$Hz is fast enough, then y can be considered an accurate representation of $y(t)$. For all the signals in this exercise, **y** can be assumed to be an accurate representation of $y(t)$. To begin this exercise, you must first load a sample audio signal by typing

>> l*oad splat*

>> y = y(1:8192));

To verify that you have accurately loaded the audio data, and to convince yourself that the MATLAB vector y can accurately represent an audio signal, type

The function **fft** takes the sampled representation **y** and computes an approximation to the CTFT of $y(t)$, $Y(j\omega)$, at samples of ω. If you type

>> Y= fftshift(fft(y));

then the vector **Y** contains an approximation to $Y(j\omega)$ at **N** evenly spaced frequencies on the interval $-\pi f_s \leq \omega < \pi f_s$. In fact, **Y** contains only approximate values of $cY(j\omega)$, where c is a constant, but you should not worry about the approximation or the scaling for the purposes of this exercise. Refer to Exercise 4.2 for a more complete discussion of the relationship between $Y(j\omega)$ and Y. The function **fftshift** re-orders the output of **fft** so that the samples of $Y(j\omega)$ are ordered in **Y** from the most negative to most positive frequencies. Most of the properties that you associate with the CTFT can now be verified on the vector **Y**.

Basic Problems

(a). Type **Y=fftshift(fft(y))** to calculate the Fourier transform vector **Y**. The corresponding frequency values can be stored in the vector **w** by typing

>> w = [-pi:2*\pi/N:pi-pi/N]*fs;

Use **w** and **Y** to plot the magnitude of the continuous-time Fourier transform over the interval $-\pi f_s \leq \omega < \pi f_s$.

The function **ifft** is the inverse operation of **fft**. For even length vectors, **fftshift** is its own inverse. For the vector **Y**, **N = 8192** and the inverse Fourier transform can be found by typing

>>y=ifft(fftshift(Y));

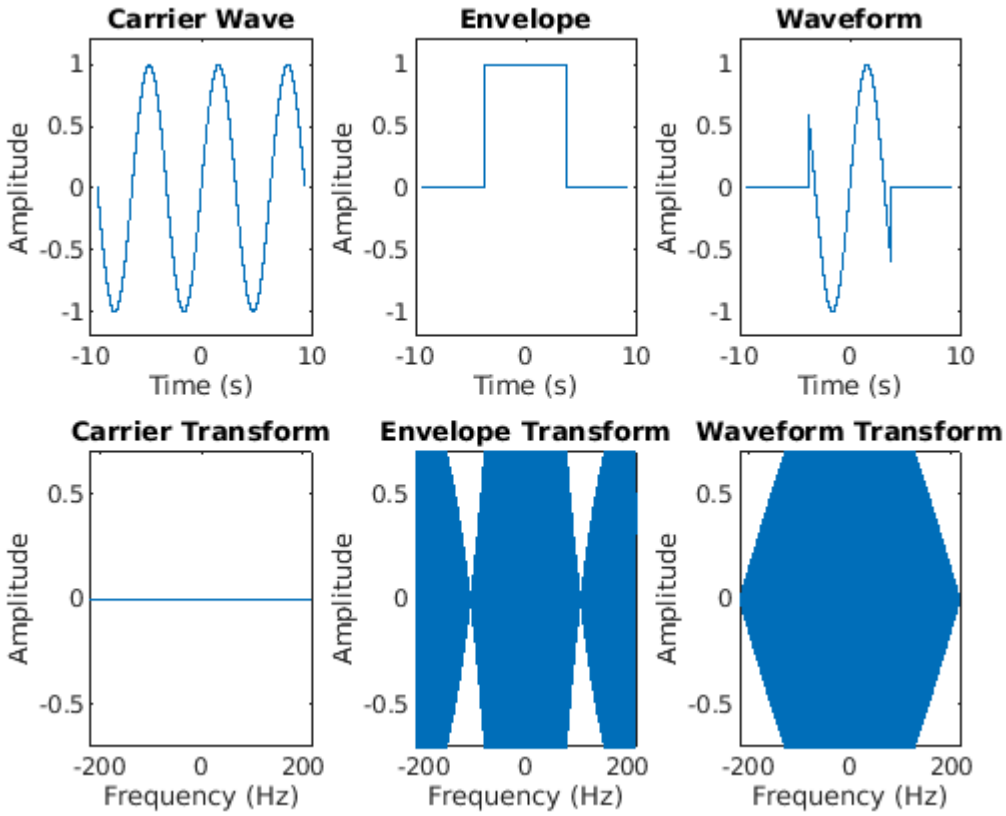
>>y=real(y);

The **real** function is applied here because the original time-domain signal was known to be real. However, numerical roundoff errors in **fft** and **ifft** can introduce a very small nonzero imaginary component to **y**. In general, the inverse CTFT is not necessarily a real signal, and the imaginary part may contain significant energy. The **real** function should only be used on the output of **ifft** when you know the resulting signal must be real, e.g., an audio signal, and you have verified the imaginary part that you will be removing is insignificant.

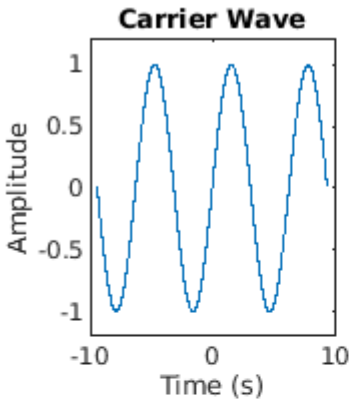
```
%4.3a

N = 8192;
fs =3;
t = [-pi:2*pi/N:pi-pi/N]*fs;
```

```
min_f = -1/(2*(t(2)-t(1)));
df = 1/(length(t)*(t(2)-t(1)));

f=min_f:df:-min_f-df;
y=zeros(1, length(t));
for i = round(0.3*length(t)):round(0.7*length(t))
    y(i)=1;
end

val_fr_crr = sin(t);
subs = y.*val_fr_crr;

val_fr_crr_fft = fftshift(fft(val_fr_crr));
y_fft = fftshift(fft(y));
subs_fft = fftshift(fft(subs));

figure;

subplot(2, 3, 1);
plot(t, val_fr_crr);
title('Carrier Wave');
ylim([-1.2 1.2]);
xlabel('Time (s)');
ylabel('Amplitude');

subplot(2, 3, 2);
plot(t, y);
title('Envelope');
ylim([-1.2 1.2]);
xlabel('Time (s)');
ylabel('Amplitude');

subplot(2, 3, 3);
plot(t, subs);
title('Waveform');
ylim([-1.2 1.2]);
xlabel('Time (s)');
ylabel('Amplitude');

subplot(2, 3, 4);
plot(f, real(val_fr_crr_fft));
title('Carrier Transform');
ylim([-0.7 0.7]);
xlabel('Frequency (Hz)');
ylabel('Amplitude');

subplot(2, 3, 5);
plot(f, real(y_fft));
title('Envelope Transform');
ylim([-0.7 0.7]);
xlabel('Frequency (Hz)');
ylabel('Amplitude');

subplot(2, 3, 6);
```

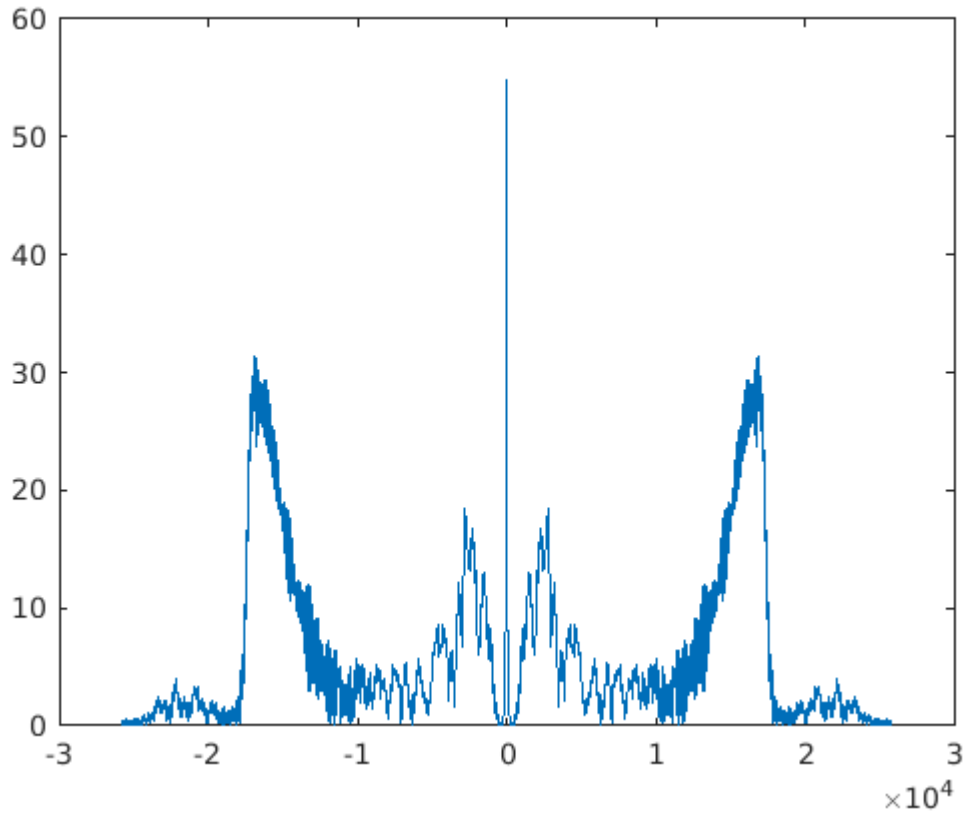
```
plot(f, real(subs_fft));
title('Waveform Transform');
ylim([-0.7 0.7]);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
```




```
figure;
subplot(2, 3, 1);
plot(t, val_fr_crr);
title('Carrier Wave');
ylim([-1.2 1.2]);
xlabel('Time (s)');
ylabel('Amplitude');
```




```
load splat;

clf;

y = y(1:8192);
N=8192;
fs = 8192;
sound(y, fs);
Y=fftshift(fft(y));

w = [-pi:2*pi/N:pi-pi/N]*fs;
plot(w, abs(Y));
```




```
y = ifft(fftshift(Y));
y = real(y);
```

(b). Set **Y1=conj (Y)**, and store in **y1** the inverse Fourier transform of **Y1**. Use **real(y1)** to ensure **y1** is real. Play **y1** using **sound(y1 ,fs)**. Can you explain what you just heard, knowing how the inverse Fourier transform of $Y*(j\omega)$ is related to $y(t)$?

```
%4.3b
Y1=conj(Y);
y1 = ifft(fftshift(Y1));
y1 = real(y1);
sound(y1, fs);
```

Intermediate Problems

The CTFT of $y(t)$ can be written in terms of its magnitude and phase as

$Y(j\omega) = |Y(j\omega)|e^{j\phi(\omega)}$,

where $\phi(\omega) = \angle Y(j\omega)$. For many signals, either the phase or the magnitude alone can be used to construct a useful approximation of the signal $y(t)$. For instance, consider the

signals $y_2(t)$ and $y_3(t)$ with CTFTs

$Y_2(j\omega) = |Y(j\omega)| \quad and \quad Y_3(j\omega) = e^{j\phi(\omega)}$

4.3b Analysis

(c). Demonstrate analytically that $y_2(t)$ and $y_3(t)$ will be real signals whenever $y(t)$ is real.

```
%4.3c Code
```

(d). Construct a vector **Y2** equal to the magnitude of **Y**. Store in **y2** the inverse Fourier transform of **Y2**. Play this vector using sound.

```
%4.3d Code
Y2 = abs(Y);
y2 = ifft(fftshift(Y2));
y2 = real(y2);
sound(y2, fs);
```

(e). Construct a vector **Y3** which has the same phase as **Y**, but has magnitude equal to one for each frequency. Store in **y3** the inverse Fourier transform of **Y3**. Play this vector using sound.

```
%4.3e Code
Y3 = angle(Y);
y3 = ifft(fftshift(Y3));
y3 = real(y3);
sound(y3, fs);
```

(f). Based upon the two signals you just heard, which component of the Fourier transform would you say is most crucial for representing an audio signal: the magnitude or the phase?

When the magnitude based signal is sounded the sound the represted original signal. When hearing the phase based signals, there is no sound. This means that the magnitude of the CTFT will hold the information of the original signal.

4.4 Time- and Frequency-Domain Characterizations of Systems

The impulse response $h(t)$ of an LTI system completely characterizes the system since the response $y(t)$ to any input $x(t)$ is given by the convolution $y(t) = h(t) * x(t)$. If the system is stable, an equivalent representation of the system is given by its frequency response $H(j\omega)$. In this case the continuous-time Fourier transforms are related by $Y(j\omega) = H(j\omega)X(j\omega)$.In this exercise, you will consider a number of stable LTI systems described by linear constant-coefficient differential equations. For these systems, you will be asked to calculate their impulse and frequency responses. Although either the frequency response or the impulse response is sufficient to completely characterize an LTI system, you will learn that it is sometimes advantageous to consider system properties in both the time and frequency domains.

Basic Problems

These problems assume that you have completed Tutorial 4.1, on f reqs. Consider the class of causal LTI systems given by

(4.11)

$$\frac{dy(t)}{dt} + a_0 y(t) = a_0 x(t)$$

where $a_0 > 0$ guarantees stability. Define System **I** to be the system satisfying Eq. (4.11) for $a_0 = 3$ and define System **II** to be the system for $a_o = 1/3.$

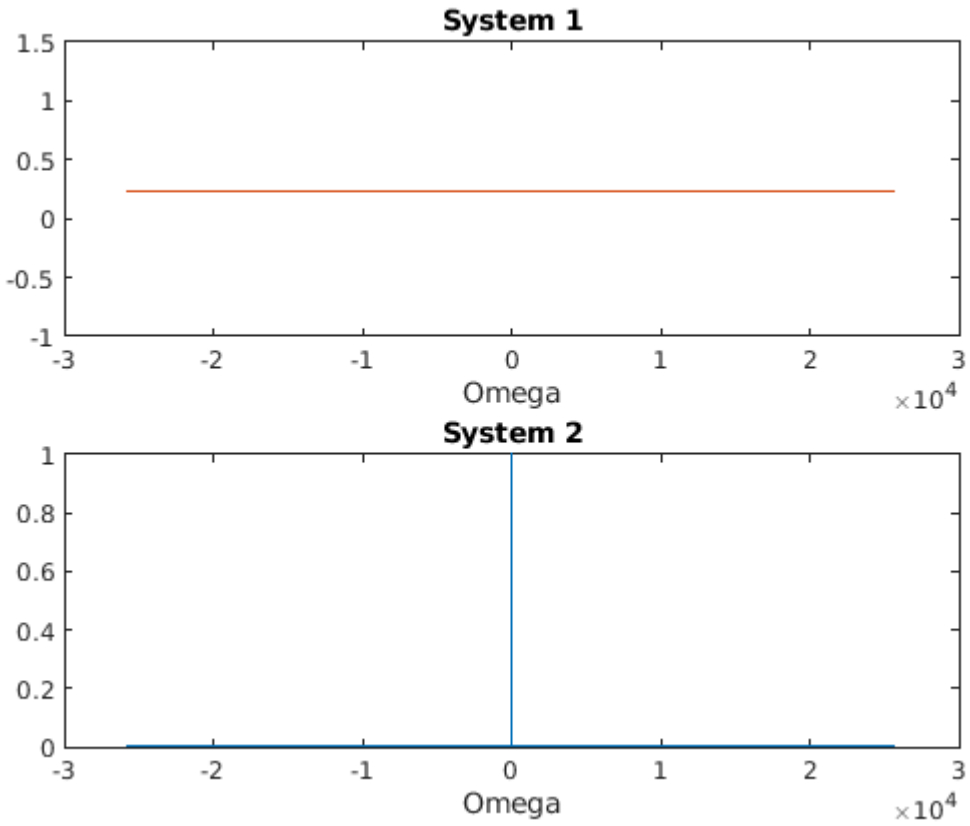
(a). Analytically derive the frequency response for the stable LTI system corresponding to Eq. (4.11). Also determine the magnitude and phase of this frequency response.

4.4a Analysis

(b). Define **w=linspace(0, 10)**. Use f reqs to calculate the frequency response of **Systems I and II** at the frequencies in w. Plot the magnitude of these two responses in a single figure. Do the magnitude plots agree with your analytic expression for the frequency response magnitude?

```
%4.4b Code

a1 = [1 3];
a1 = 13;
b1 = 3;
b1 = 3;
S1 = freqs(b1, a1, w);

a2 = [1 1/3];
b2 = [1/3];
S2 = freqs(b2, a2, w);
```

```
subplot(2, 1, 1);
plot(w, abs(S1));
title('System 1');
xlabel('Omega');

hold on;

subplot(2, 1, 2);
plot(w, abs(S2));
title('System 2');
xlabel('Omega');
```



(c). Use the function impulse described in Tutorial 3.3 to calculate the impulse response of **Systems I and II** at time samples defined by the **vector t=linspace(0,5).**

```
%4.4c Code
```

(d). What is the relationship between the rate at which the impulse response decays (with time) and the rate at which the frequency response magnitude decays (with frequency)? What CTFT property explains this relationship?

4.4d Analysis