

Lab 4: The Continuous-Time Fourier Transform

Author	Name & Student ID : 王卓扬 (12112907)、冯彦捷 (12010825)
---------------	---

Introduction

1. Using MATLAB to calculate the approximation of Continuous-time Fourier Transform.
2. Using MATLAB to do simple sampling.
3. Using *abs* and *angle* to calculate the magnitude and the phase of the signal.
4. Using MATLAB to conduct Amplitude Modulation.
5. Using *freqs* to calculate the frequency response of a LTI system described by the linear-constant-coefficient differential equation.
6. Using *lsim* to calculate the response of a LTI system described by the linear-constant-coefficient differential equation.
7. Recognizing the Morse Code.

Lab results & Analysis:

■ 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

$$\int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt = \lim_{\tau \rightarrow 0} \sum_{n=-\infty}^{\infty} x(n\tau)e^{-j\omega n\tau} \tau.$$

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

$$\int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt = \int_0^T x(t)e^{-j\omega t} dt \approx \sum_{n=0}^{N-1} x(n\tau)e^{-j\omega n\tau} \tau, \quad (4.7)$$

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 \mathbf{x} then the function call `X=tau*fft(x)` calculates

$$\mathbf{X}(k+1) = \tau \sum_{n=0}^{N-1} x(n\tau)e^{-j\omega_k n\tau}, \quad 0 \leq k < N, \quad (4.8)$$

$$\approx X(j\omega_k), \quad (4.9)$$

where

$$\omega_k = \begin{cases} \frac{2\pi k}{N\tau}, & 0 \leq k \leq \frac{N}{2}, \\ \frac{2\pi k}{N\tau} - \frac{2\pi}{\tau}, & \frac{N}{2} + 1 \leq k < N, \end{cases} \quad (4.10)$$

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 \mathbf{X} samples of $X(j\omega_k)$ ordered such that $\mathbf{X}(k+1)$ is the CTFT evaluated at $-\pi/\tau + (2\pi k/N\tau)$ for $0 \leq k \leq N-1$, 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 $g(t) = e^{-2t}u(t)$.

Solution (a):

$$X(j\omega) = \frac{1}{2 + j\omega} + \frac{1}{2 - j\omega}$$

Analysis (a):

Let

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

The Fourier Transform for $g(t)$ is

$$G(j\omega) = \frac{1}{2 + j\omega}$$

According to the properties we have

$$X(j\omega) = G(j\omega) + G(-j\omega) = \frac{1}{2 + j\omega} + \frac{1}{2 - j\omega}$$

- (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=[0: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 .

Solution (b):

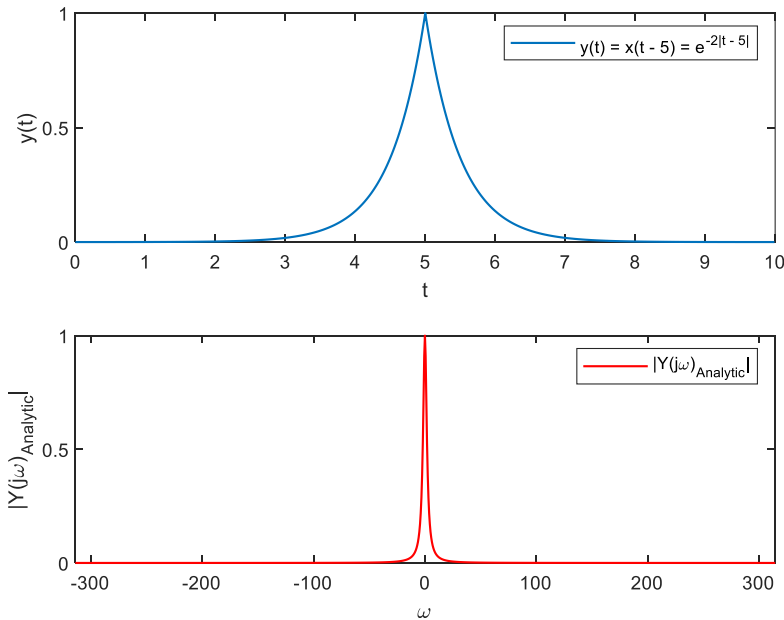


Fig 4.2.b

Analysis (b): Vector `y` and analytically calculated $|Y(j\omega)|$ are plotted in Fig 4.2.b. And

$$Y(j\omega) = e^{j\omega(-5)}X(j\omega) = \frac{e^{j\omega(-5)}}{2 + j\omega} + \frac{e^{j\omega(-5)}}{2 - j\omega}$$

(c). Calculate samples $Y(j\omega_k)$ by typing `Y=fftshift(tau*fft(y))`.

Solution (c):

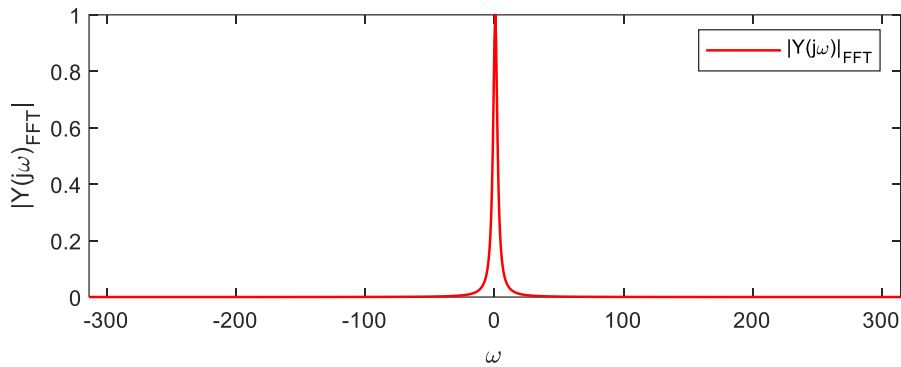


Fig 4.2.c

Analysis (c): The amplitude of the Fourier Transform for $y(t)$ using function `fft` is shown in Fig 4.2.c.

(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));
```

Solution (d):

We have defined in the program for (a).

Analysis (d): None.

(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`.

Solution (e):

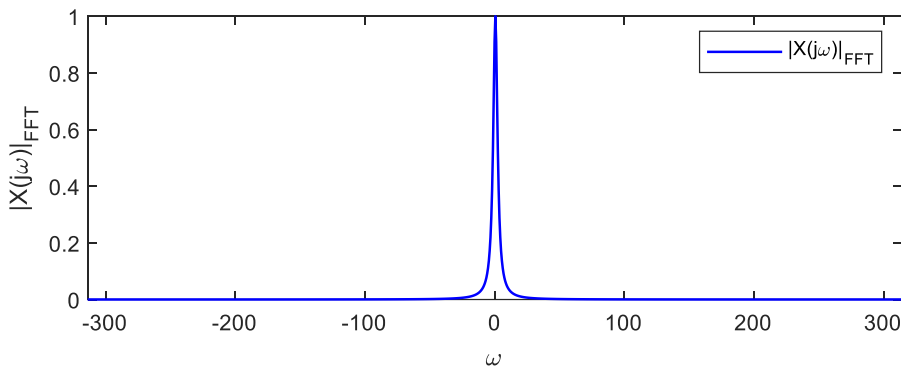


Fig 4.2.e

Analysis (e): $X(j\omega)$ is plotted in Fig 4.2.e according to the formula below.

$$\mathcal{F}\{x(t)\} = e^{j5t}\mathcal{F}\{x(t-5)\} = e^{j5t}\mathcal{F}\{y(t)\}$$

(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 .

Solution (f):

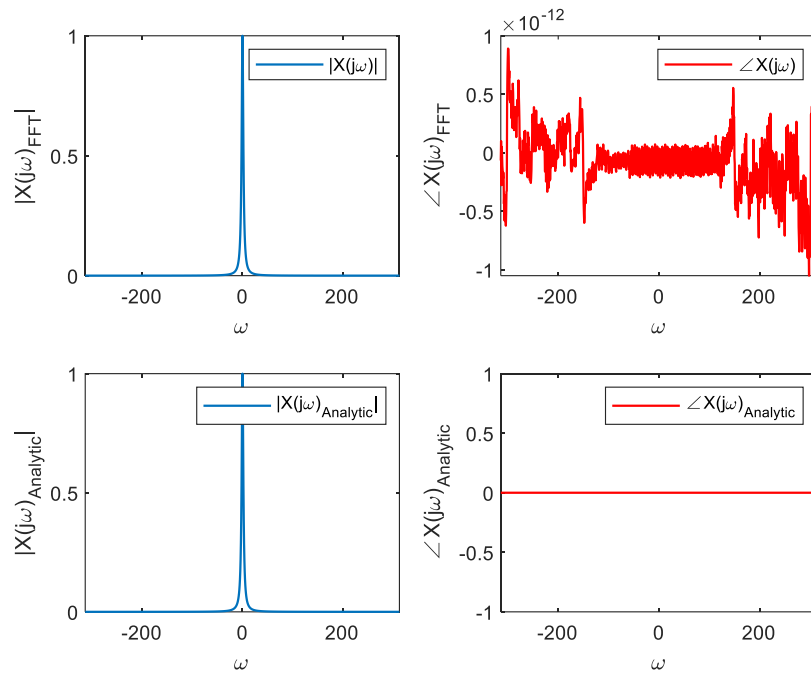


Fig 4.2.f (1)

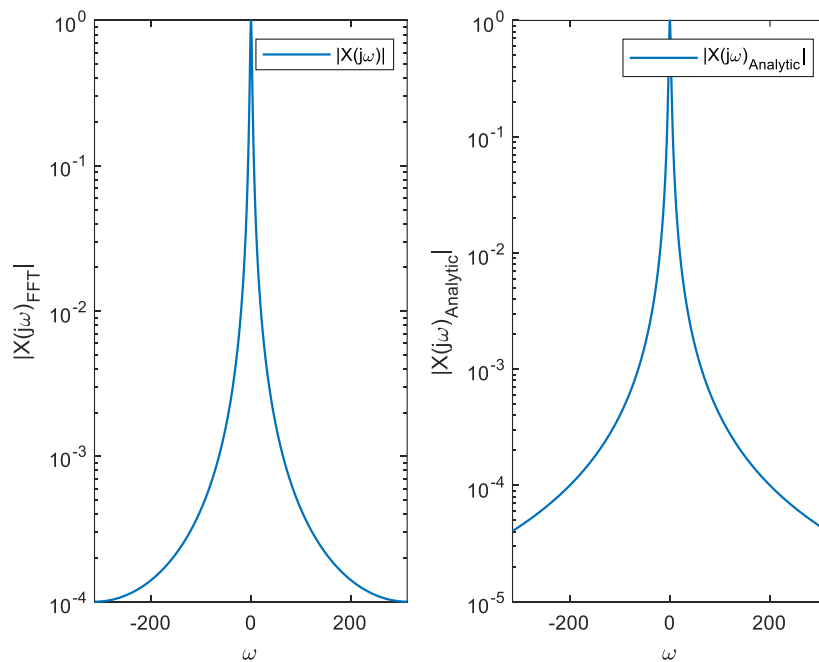


Fig 4.2.f (2)

Analysis (f): The result is shown in Fig 4.2.f (1). We can see that the magnitude of the approximation and the analytic result is equal, as well as the angle (error is small enough). In Fig 4.2.f (2) is the magnitude plotted by function *semilogy*, in which we see that they are not identical at the high frequencies.

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

Solution (g):

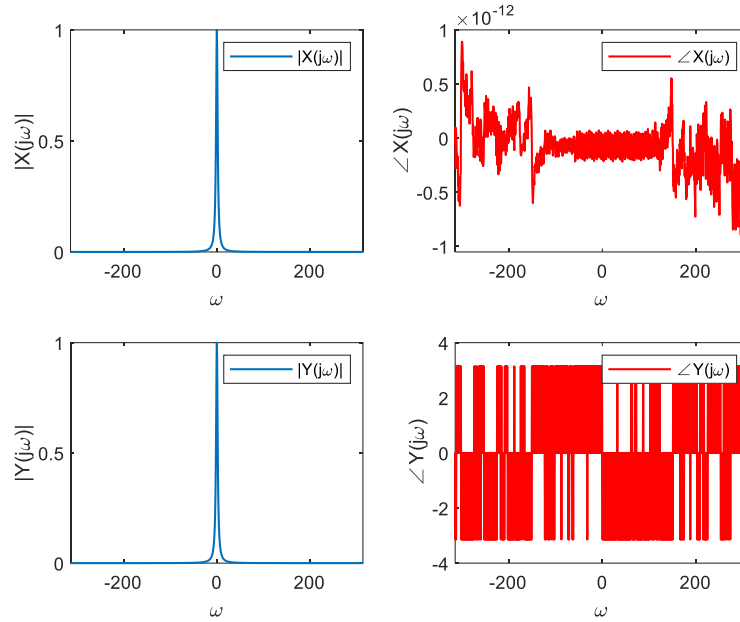


Fig 4.2.g

Analysis (g): In Fig 4.2.g are the patterns for X and Y. They are the same in magnitude but distinct in phase. That is because the time shifting was reflected as a phase shifting in frequency domain.

■ 4.6 Amplitude Modulation and the Continuous-Time Fourier Transform

This exercise will explore amplitude modulation of Morse code messages. A simple amplitude modulation system can be described by

$$x(t) = m(t) \cos(2\pi f_0 t), \quad (4.13)$$

where $m(t)$ is called the message waveform and f_0 is the modulation frequency. The continuous-time Fourier transform (CTFT) of a cosine of frequency f_0 is

$$C(j\omega) = \pi\delta(\omega - 2\pi f_0) + \pi\delta(\omega + 2\pi f_0), \quad (4.14)$$

which can be confirmed by substituting $C(j\omega)$ into Eq. (4.1) to yield

$$\cos(2\pi f_0 t) = \frac{1}{2} (e^{j2\pi f_0 t} + e^{-j2\pi f_0 t}). \quad (4.15)$$

Using $C(j\omega)$ and the multiplication property of the CTFT, you can obtain the CTFT of $x(t)$,

$$X(j\omega) = \frac{1}{2} M(j(\omega - 2\pi f_0)) + \frac{1}{2} M(j(\omega + 2\pi f_0)), \quad (4.16)$$

where $M(j\omega)$ is the CTFT of $m(t)$. Since the CTFT of a sinusoid can be expressed in terms of impulses in the frequency domain, multiplying the signal $m(t)$ by a cosine places copies of $M(j\omega)$ at the modulation frequency.

The remainder of this exercise will involve the signal,

$$x(t) = m_1(t) \cos(2\pi f_1 t) + m_2(t) \sin(2\pi f_2 t) + m_3(t) \sin(2\pi f_1 t), \quad (4.17)$$

and several parameters that can be loaded into MATLAB from the file `ctftmod.mat`. This file is in the Computer Explorations Toolbox, which can be obtained from The MathWorks at the address listed in the Preface. If the file is in one of the directories in your MATLAB-PATH, type `load ctftmod.mat` to load the required data. The directories contained in your MATLABPATH can be listed by typing `path`. If the file has been successfully loaded, then typing `who` should produce the following result:

```
>> who
```

Your variables are:

```
af      dash      f1      t
bf      dot       f2      x
```

In addition to the signal $x(t)$, you also have loaded:

- a lowpass filter, whose frequency response can be plotted by `freqs(bf,af)`,
- modulation frequencies `f1` and `f2`,
- two prototype signals `dot` and `dash`,
- a sequence of time samples `t`.

To make this exercise interesting, the signal $x(t)$ contains a simple message. When loading the file, you should have noticed that you have been transformed into Agent 008, the code-breaking sleuth. The last words of the aging Agent 007 were “The future of technology lies in ...” at which point Agent 007 produced a floppy disk and keeled over. The floppy disk contained the MATLAB file `ctftmod.mat`. Your job is to decipher the message encoded in $x(t)$ and complete Agent 007’s prediction.

Here is what is known. The signal $x(t)$ is of the form of Eq. (4.17), where f_1 and f_2 are given by the variables `f1` and `f2`, respectively. It is also known that each of the signals $m_1(t)$, $m_2(t)$ and $m_3(t)$ correspond to a single letter of the alphabet which has been encoded using International Morse Code, as shown in the following table:

A	.-	H	O	---	V	...-
B	-...	I	..	P	W	---
C	----	J	----	Q	----	X	---
D	-..	K	---	R	---	Y	----
E	.	L	S	...	Z	----
F	M	--	T	-		
G	---	N	-.	U	..		

Basic Problems

- (a). Using the signals `dot` and `dash`, construct the signal that corresponds to the letter ‘Z’ in Morse code, and plot it against `t`. As an example, the letter C is constructed by typing `c = [dash dot dash dot]`. Store your signal $z(t)$ in the vector `z`.

Solution (a):

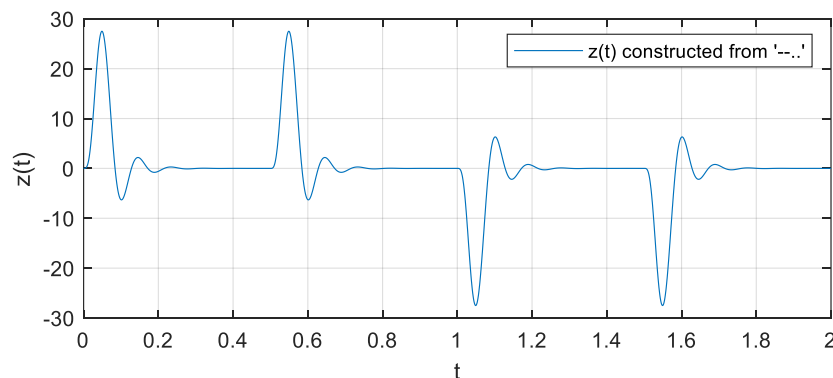


Fig 4.6.a

Analysis (a): $z(t)$ constructed from “---.” with *dash* and *dot* is shown in Fig 4.6.a.

- (b). Plot the frequency response of the filter using `freqs(bf,af)`.

Solution (b):

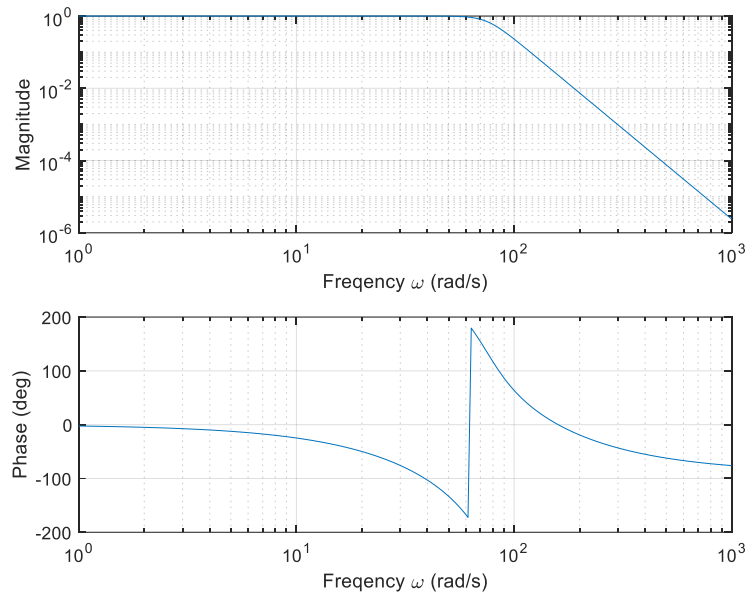


Fig 4.6.b

Analysis (b): The magnitude and phase of the frequency response plotted by *freqs(bf, af)* are shown in Fig 4.6.b.

- (c). The signals **dot** and **dash** are each composed of low frequency components such that their Fourier transforms lie roughly within the passband of the lowpass filter. Demonstrate this by filtering each of the two signals using

```
>> ydash=lsim(bf,af,dash,t(1:length(dash)));
>> ydot=lsim(bf,af,dot,t(1:length(dot)));
```

Plot the outputs **ydash** and **ydot** along with the original signals **dash** and **dot**.

Solution (c):

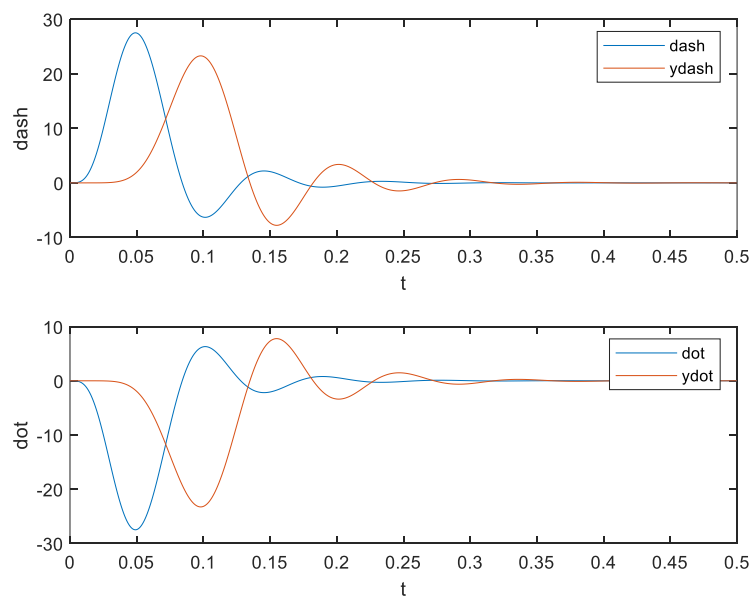


Fig 4.6.c

Analysis (c): From the result in Fig 4.6.c we can find that most of the energy of dot signal and dash signal can successfully pass the filter (the shape doesn't change too much).

- (d). When the signal `dash` is modulated by $\cos(2\pi f_1 t)$, most of the energy in the Fourier transform will move outside the passband of the filter. Create the signal $y(t)$ by executing `y=dash*cos(2*pi*f1*t(1:length(dash)))`. Plot the signal $y(t)$. Also plot the output `yo=lsim(bf,af,y,t)`. Do you get a result that you would have expected?

Solution (d):

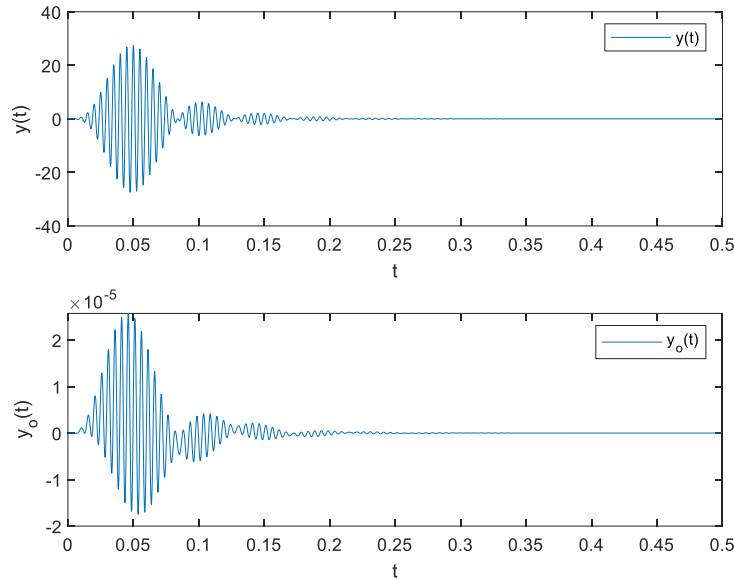


Fig 4.6.d

Analysis (d): According to the changes in magnitude of signal in Fig 4.6.d, we can find that most of the energy failed in passing the filter, as what we expected.

Intermediate Problems

- (e). Determine analytically the Fourier transform of each of the signals

$$m(t) \cos(2\pi f_1 t) \cos(2\pi f_1 t),$$

$$m(t) \cos(2\pi f_1 t) \sin(2\pi f_1 t),$$

and

$$m(t) \cos(2\pi f_1 t) \cos(2\pi f_2 t),$$

in terms of $M(j\omega)$, the Fourier transform of $m(t)$.

Solution (e):

The results are here:

$$Y_1(j\omega) = \frac{1}{4} M[j(\omega - 4\pi f_1)] + \frac{1}{2} M(j\omega) + \frac{1}{4} M[j(\omega + 4\pi f_1)]$$

$$Y_2(j\omega) = \frac{1}{4j} M[j(\omega - 4\pi f_1)] - \frac{1}{4j} M[j(\omega + 4\pi f_1)]$$

$$Y_3(j\omega) = \frac{1}{4} M[j(\omega - 2\pi f_1 - 2\pi f_2)] + \frac{1}{4} M[j(\omega - 2\pi f_1 + 2\pi f_2)] \\ + \frac{1}{4} M[j(\omega + 2\pi f_1 - 2\pi f_2)] + \frac{1}{4} M[j(\omega + 2\pi f_1 + 2\pi f_2)]$$

Analysis (e): Here's the derivation process:

$$Y_1(j\omega) = \frac{1}{2} \left(\frac{1}{2} M[j(\omega - 4\pi f_1)] + \frac{1}{2} M(j\omega) \right) + \frac{1}{2} \left(\frac{1}{2} M(j\omega) + \frac{1}{2} M[j(\omega + 4\pi f_1)] \right) = \dots$$

$$Y_2(j\omega) = \frac{1}{2j} \left(\frac{1}{2} M[j(\omega - 4\pi f_1)] + \frac{1}{2} M(j\omega) \right) - \frac{1}{2j} \left(\frac{1}{2} M(j\omega) + \frac{1}{2} M[j(\omega + 4\pi f_1)] \right) = \dots$$

$$Y_3(j\omega) = \frac{1}{2} \left(\frac{1}{2} M[j(\omega - 2\pi f_1 - 2\pi f_2)] + \frac{1}{2} M[j(\omega - 2\pi f_1 + 2\pi f_2)] \right) + \frac{1}{2} \left(\frac{1}{2} M[j(\omega + 2\pi f_1 - 2\pi f_2)] + \frac{1}{2} M[j(\omega + 2\pi f_1 + 2\pi f_2)] \right) = \dots$$

- (f). Using your results from Part (e) and by examining the frequency response of the filter as plotted in Part (b), devise a plan for extracting the signal $m_1(t)$ from $x(t)$. Plot the signal $m_1(t)$ and determine which letter is represented in Morse code by the signal.

Solution (f):

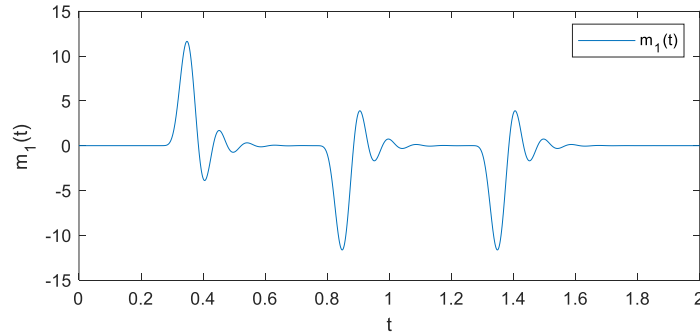


Fig 4.6.f

The pattern represents “-..”.

According to the given table we can figure out that the letter 1 is “D”.

Analysis (f):

According to the results in Part (e), when we multiply $x(t)$ by $\cos(2\pi f_1 t)$, the Fourier Transform of the result $x'(t)$ will have a $\frac{1}{2} M_1(j\omega)$ on the low frequency band, while other components are on higher frequency band. Then we take it as the input of the given low-pass filter and the frequency components of $\frac{1}{2} m_1(t)$ will remain, while others may be filtered out.

- (g). Repeat Part (f) for the signals $m_2(t)$ and $m_3(t)$. Agent 008, where does the future of technology lie?

Solution (g):

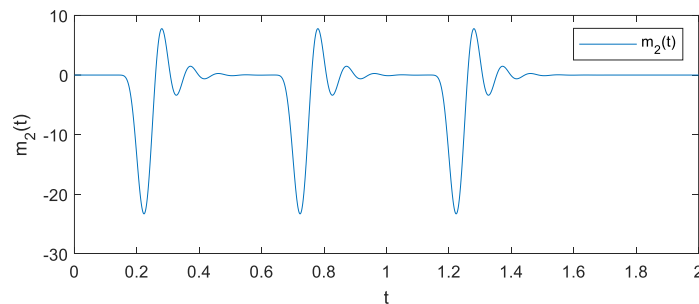


Fig 4.6.g (1)

The pattern represents “...”.

According to the given table we can figure out that the letter 1 is “S”.

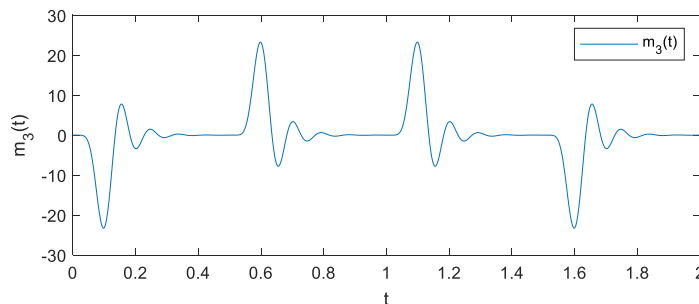
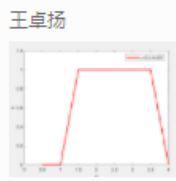
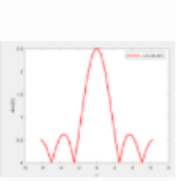
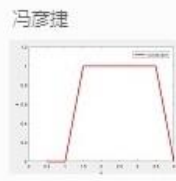
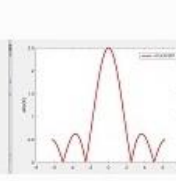
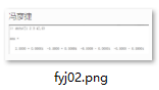

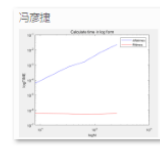
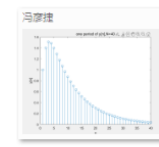
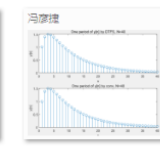
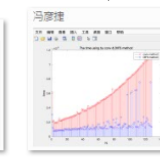
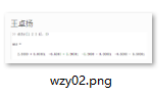
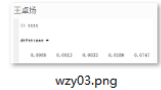
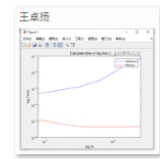
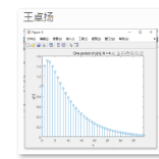
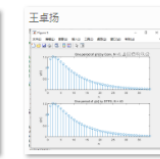
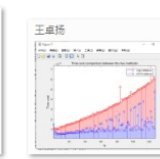


Fig 4.6.g (2)

<p><u>The pattern represents “.-.”</u> <u>According to the given table we can figure out that the letter 1 is “P”.</u> <u>So the message is “DSP”.</u> Analysis (g): “DSP” may mean “Digital Signal Processing”?</p>	
<p>Experience</p> <ol style="list-style-type: none"> 1. This time we didn’t use the util class. We think we need to practice the ability to code in a plain method in order to prepare for the code test. 2. Read the questions carefully! We didn’t notice the meaning of the given variable t and were puzzled to the problem and wasted too much time. 3. Problem 4.6 is interesting. 4. In the calculation of CTFT, the parity of N is important and it will affect the sampling of omega! <p>* The screenshots in lab class:</p> <p style="text-align: center;">Last last week:</p> <div style="display: flex; justify-content: space-around;">     </div> <p style="text-align: center;">And last week (we are not sure if you said these are needed to be submitted):</p> <div style="display: grid; grid-template-columns: repeat(6, 1fr); gap: 10px;">             </div>	
Score	99

Appendix

Programs for 4.2 (All-in-one)

```
clear;

%% (b)
tau = 0.01;
T = 10;
N = T / tau;

t = linspace(0, T - tau, N);
w = -(pi / tau) + (0 : N - 1) * (2 * pi / (N * tau)); % Even N
y = exp(-2 * abs(t - 5));
```

```

Xa = 1 ./ (2 + 1j .* w) + 1 ./ (2 - 1j .* w);
Ya = exp(-1j * 5 * w) .* Xa;

figure("Name", "b");
subplot(2, 1, 1), plot(t, y, "LineWidth", 1);
xlabel("t"), ylabel("y(t)");
legend("y(t) = x(t - 5) = e^{-2|t - 5|}");
subplot(2, 1, 2), plot(w, abs(Ya), "r", "LineWidth", 1);
xlabel("\omega"), ylabel("|Y(j\omega)_{Analytic}|"), xlim([w(1), w(end)]);
legend("|Y(j\omega)_{Analytic}|");

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

figure("Name", "c");
plot(w, abs(Y), "r", "LineWidth", 1);
xlabel("\omega"), ylabel("|Y(j\omega)_{FFT}|"), xlim([w(1), w(end)]);
legend("|Y(j\omega)_{FFT}|");

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

figure("Name", "e");
plot(w, abs(X), "b", "LineWidth", 1);
xlabel("\omega"), ylabel("|X(j\omega)_{FFT}|"), xlim([w(1), w(end)]);
legend("|X(j\omega)_{FFT}|");

%% (f) - 1
figure("Name", "f1");
subplot(2, 2, 1), plot(w, abs(X), "LineWidth", 1);
xlabel("\omega"), ylabel("|X(j\omega)_{FFT}|"), xlim([w(1), w(end)]);
legend("|X(j\omega)|");
subplot(2, 2, 2), plot(w, angle(X), "r", "LineWidth", 1);
xlabel("\omega"), ylabel("\angle X(j\omega)_{FFT}"), xlim([w(1), w(end)]);
legend("\angle X(j\omega)");
subplot(2, 2, 3), plot(w, abs(Xa), "LineWidth", 1);
xlabel("\omega"), ylabel("|X(j\omega)_{Analytic}|"), xlim([w(1), w(end)]);
legend("|X(j\omega)_{Analytic}|");
subplot(2, 2, 4), plot(w, angle(Xa), "r", "LineWidth", 1);
xlabel("\omega"), ylabel("\angle X(j\omega)_{Analytic}"), xlim([w(1), w(end)]);
legend("\angle X(j\omega)_{Analytic}");

%% (f) - 2
figure("Name", "f2");

```

```

subplot(1, 2, 1), semilogy(w, abs(X), "LineWidth", 1);
xlabel("\omega"), ylabel("|X(j\omega)_{FFT}|"), xlim([w(1), w(end)]);
legend("|X(j\omega)|");
subplot(1, 2, 2), semilogy(w, abs(Xa), "LineWidth", 1);
xlabel("\omega"), ylabel("|X(j\omega)_{Analytic}|"), xlim([w(1), w(end)]);
legend("|X(j\omega)_{Analytic}|");

%% (g)
figure("Name", "g");
subplot(2, 2, 1), plot(w, abs(X), "LineWidth", 1);
xlabel("\omega"), ylabel("|X(j\omega)|"), xlim([w(1), w(end)]);
legend("|X(j\omega)|");
subplot(2, 2, 2), plot(w, angle(X), "r", "LineWidth", 1);
xlabel("\omega"), ylabel("\angle X(j\omega)"), xlim([w(1), w(end)]);
legend("\angle X(j\omega)");
subplot(2, 2, 3), plot(w, abs(Y), "LineWidth", 1);
xlabel("\omega"), ylabel("|Y(j\omega)|"), xlim([w(1), w(end)]);
legend("|Y(j\omega)|");
subplot(2, 2, 4), plot(w, angle(Y), "r", "LineWidth", 1);
xlabel("\omega"), ylabel("\angle Y(j\omega)"), xlim([w(1), w(end)]);
legend("\angle Y(j\omega)");

```

Programs for 4.6 (All-in-one)

```

clear;

%% (a)
load("ctftmod.mat");
z = [dash, dash, dot, dot];

figure("Name", "a");
plot(t, z), xlabel("t"), ylabel("z(t)");
legend("z(t) constructed from '--..');
grid on;

%% (b)
figure("Name", "b");
freqs(bf, af);

%% (c)
ydash = lsim(bf, af, dash, t(1 : length(dash)))';
ydot = lsim(bf, af, dot, t(1 : length(dot)))';

figure("Name", "c");

```

```

subplot(2, 1, 1), plot(t(1 : length(dash)), dash, t(1 : length(ydash)),
ydash);
xlabel("t"), ylabel("dash"), legend("dash", "ydash");
subplot(2, 1, 2), plot(t(1 : length(dot)), dot, t(1 : length(ydot)), ydot);
xlabel("t"), ylabel("dot"), legend("dot", "ydot");

```

%% (d)

```

y = dash .* cos(2 * pi * f1 * t(1 : length(dash)));
yo = lsim(bf, af, y, t(1 : length(y)));

```

```

figure("Name", "d");
subplot(2, 1, 1), plot(t(1 : length(y)), y);
xlabel("t"), ylabel("y(t)"), legend("y(t)");
subplot(2, 1, 2), plot(t(1 : length(yo)), yo);
xlabel("t"), ylabel("y_o(t)"), legend("y_o(t)");

```

%% (f)

```

x1 = x .* cos(2 * pi * f1 * t);
m1 = 2 * lsim(bf, af, x1, t);

```

```

figure("Name", "f");
plot(t, m1), xlabel("t"), ylabel("m_1(t)");
legend("m_1(t)");

```

%% (g) - 1

```

x2 = x .* sin(2 * pi * f2 * t);
m2 = 2 * lsim(bf, af, x2, t);

```

```

figure("Name", "g1");
plot(t, m2), xlabel("t"), ylabel("m_2(t)");
legend("m_2(t)");

```

%% (g) - 2

```

x3 = x .* sin(2 * pi * f1 * t);
m3 = 2 * lsim(bf, af, x3, t);

```

```

figure("Name", "g2");
plot(t, m3), xlabel("t"), ylabel("m_3(t)");
legend("m_3(t)");

```