LAB4 MATLAB Programming

涂峻绫(12213010) and 欧阳安男(12211831)

Contents

Introduction

This lab is to use MATLAB to compute numerical approximations to the CTFT integral, and explore amplitude modulation of Morse code messages.

Results and Analysis

4.2

(a)

■ 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

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

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

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

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 **x** then the function call **X=tau*fft(x)** calculates

$$\begin{aligned} \mathtt{X}(k+1) &= \tau \sum_{n=0}^{N-1} x(n\tau)e^{-j\omega_k n\tau}, \qquad 0 \leq k < N, &(4.8)\\ &\approx X(j\omega_k), &(4.9)\end{aligned}$$

where

$$\omega_k = \begin{cases} \dfrac{2\pi k}{N\tau}, & 0 \leq k \leq \dfrac{N}{2}, \\ \dfrac{2\pi k}{N\tau} - \dfrac{2\pi}{\tau}, & \dfrac{N}{2} + 1 \leq k < N, \end{cases} \tag{4.10}$$

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 **X** samples of $X(j\omega_k)$ ordered such that $\mathtt{X}(k+1)$ 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 $g(t) = e^{-2t}u(t)$.

Initialize.

```
clear; clc; close all;
```

Calculate the analytic expression for $x(t)$.

$$\begin{aligned} G(j\omega) &= F\{g(t)\} \\ &= \frac{1}{2+j\omega} \end{aligned}$$

$$\begin{aligned} X(j\omega) &= F\{x(t)\} \\ &= F\{g(t)+g(-t)\} \\ &= F\{g(t)\}+F\{g(-t)\} \\ &= G(j\omega)+G(-j\omega) \\ &= \frac{1}{2+j\omega}+\frac{1}{2-j\omega} \end{aligned}$$

(b)

(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.

Create τ, T, N, vector t_x, x, t, y.

```
tau = 0.01;
T = 10;
N = T/tau;
t = 0:tau:T-tau;
tx = t-5;
x = exp(-2*abs(tx));
y = x;
```

(c)

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

Calculate CTFT of $y(t) = x(t-5)$.

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

(d)

(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));
```

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

(e)

(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**.

```
X = Y.*exp(1i*5*w);
```

(f)

(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.

Calculate X_{abs}, X_{angle}.

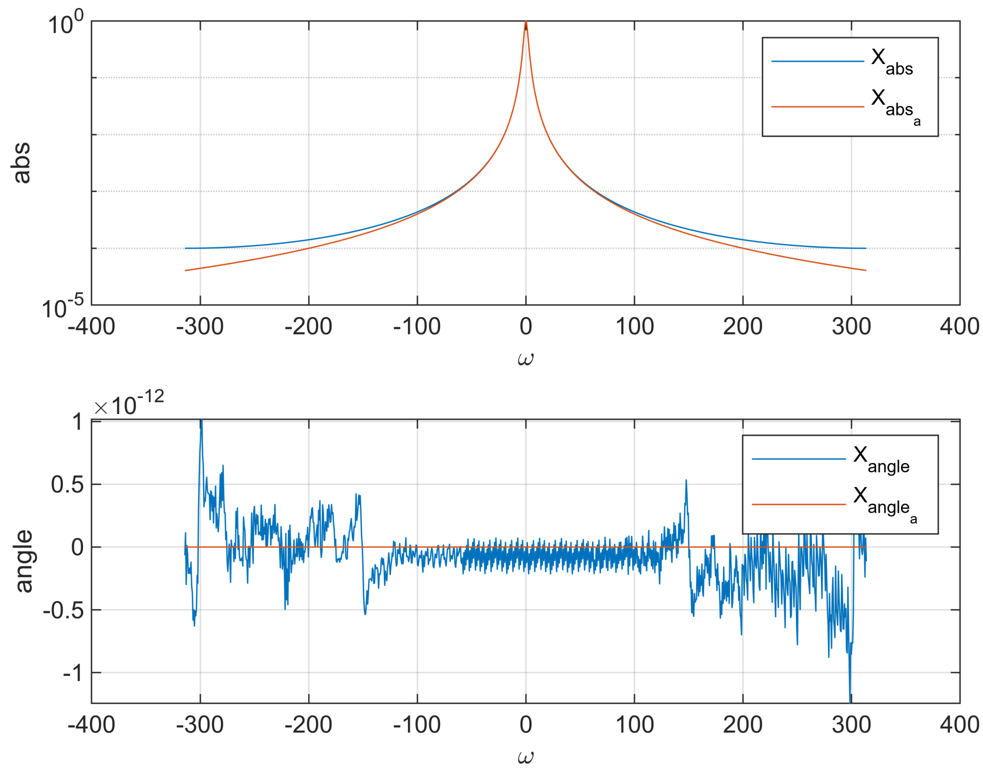
```
Xabs = abs(X);
Xangle = angle(X);
```

Calculate X_a, X_{abs_a}, X_{angle_a}.

```
X_a = 1./(2+1i*w)+1./(2-1i*w);
Xabs_a = abs(X_a);
Xangle_a = angle(X_a);
```

Plot X_{abs}, X_{abs_a}, X_{angle}, X_{angle_a}.

```
figure;
subplot(2,1,1);
semilogy(w,Xabs,w,Xabs_a);
legend("X_{abs}","X_{abs_a}");
grid on;
xlabel("\omega");
ylabel("abs");
subplot(2,1,2);
plot(w,Xangle,w,Xangle_a);
legend("X_{angle}","X_{angle_a}");
grid on;
xlabel("\omega");
ylabel("angle");
```



The approximation is totally matches when the frequency is low. But when the frequency is high, the approximation is not such accurate.

(g)

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

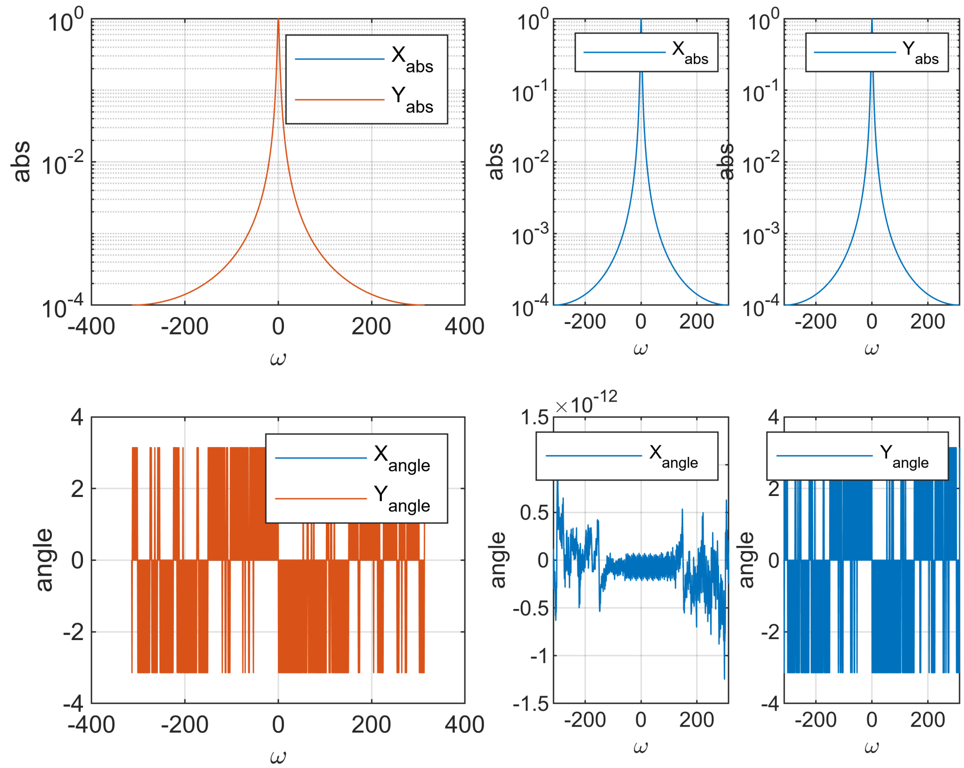
Calculate Y_{abs}, Y_{angle}.

```
Yabs = abs(Y);
Yangle = angle(Y);
```

Plot X_{abs}, Y_{abs}, X_{angle}, Y_{angle}.

```
figure;
subplot(2,2,1);
semilogy(w,Xabs,w,Yabs);
legend("X_{abs}","Y_{abs}");
grid on;
xlabel("\omega");
ylabel("abs");
subplot(2,2,3);
plot(w,Xangle,w,Yangle);
legend("X_{angle}","Y_{angle}");
```

```
grid on;
xlabel("\omega");
ylabel("angle");
subplot(2,4,3);
semilogy(w,Xabs);
legend("X_{abs}");
grid on;
xlabel("\omega");
ylabel("abs");
subplot(2,4,4);
semilogy(w,Yabs);
legend("Y_{abs}");
grid on;
xlabel("\omega");
ylabel("abs");
subplot(2,4,7);
plot(w,Xangle);
legend("X_{angle}");
grid on;
xlabel("\omega");
ylabel("angle");
subplot(2,4,8);
plot(w,Yangle);
legend("Y_{angle}");
grid on;
xlabel("\omega");
ylabel("angle");
```



X_{abs} and Y_{abs} are the same while X_{angle} and Y_{angle} are very different. It's because the time shift only leads to phase shift in frequency domain, which is also mentioned in **(e)**.

4.6

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

$$x(t) = m(t)\cos(2\pi f_0 t), \tag{4.13}$$

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

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

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

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

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

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

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,

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

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	·· -		

```
clc;clear;close all;
```

Load the file *ctftmod.mat*.

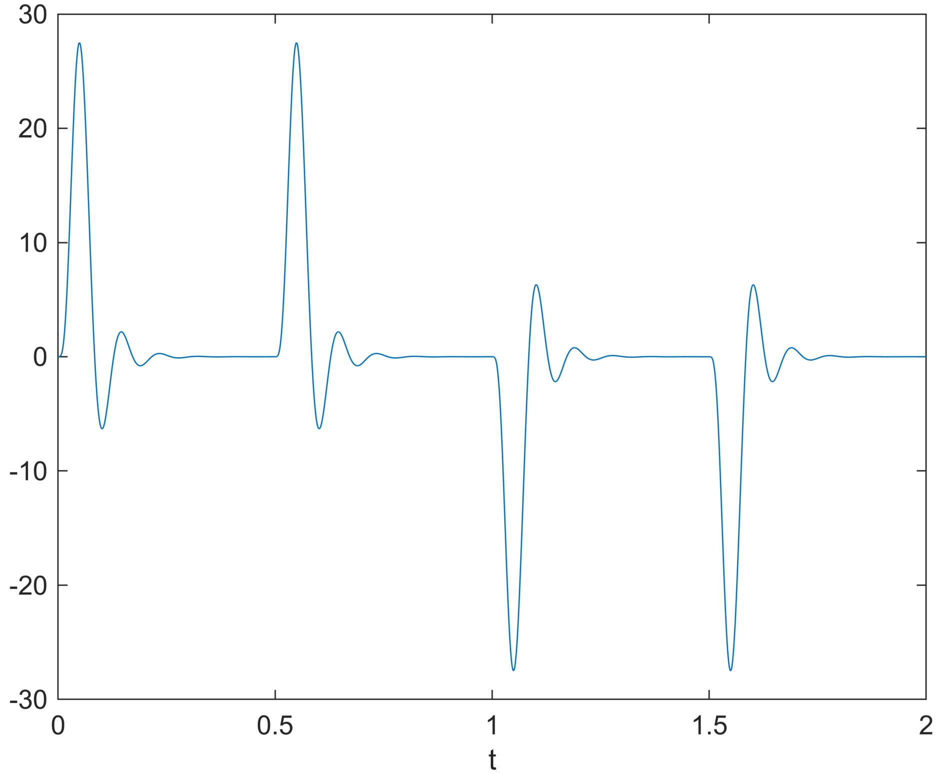
```
load ctftmod.mat
```

(a)

(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**.

Create the signal of Z in Morse code, plot it against t, and store the signal in the vector z.

```
z = [dash dash dot dot];
figure;
plot(t,z);xlabel("t");
```

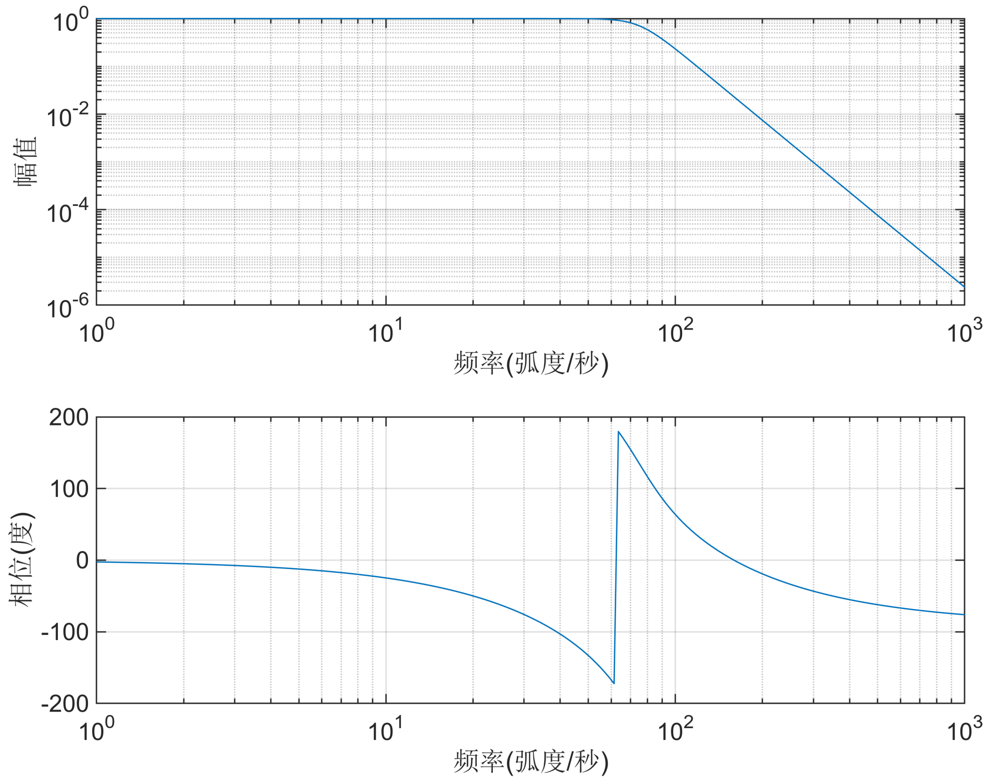


(b)

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

Plot the frequency response of the lowpass filter.

```
freqs(bf,af);
```



(c)

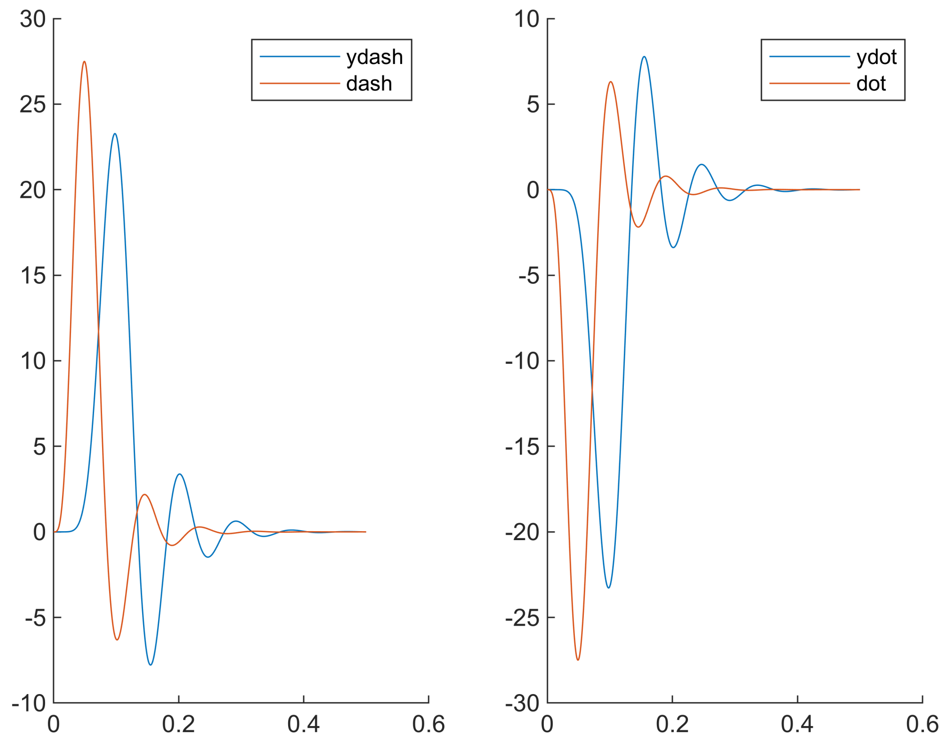
(c). The signals **dot** and **dash** are each composed of low frequency components such that their Fourier transforms lie roughly within the passband of the lowpass filter. Demonstrate this by filtering each of the two signals using

```
>> 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**.

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

```
ydash = lsim(bf,af,dash,t(1:length(dash)));
ydot = lsim(bf,af,dot,t(1:length(dot)));
figure;
subplot(1,2,1);hold on;
plot(t(1:length(ydash)),ydash);
plot(t(1:length(dash)),dash);
legend('ydash','dash');
subplot(1,2,2);hold on;
plot(t(1:length(ydot)),ydot);
plot(t(1:length(dot)),dot);
legend('ydot','dot');
```

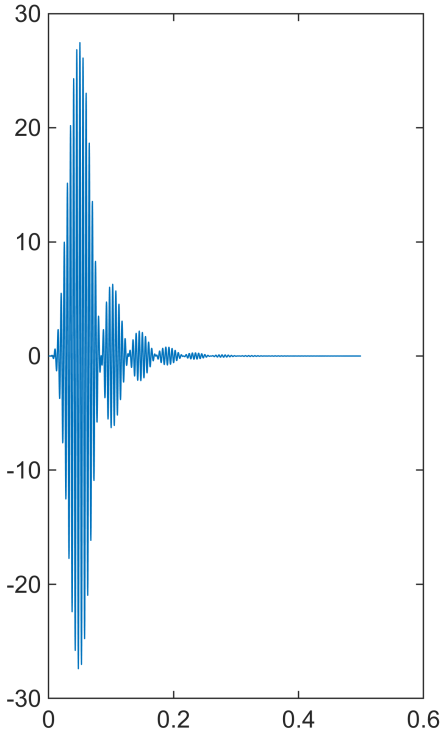


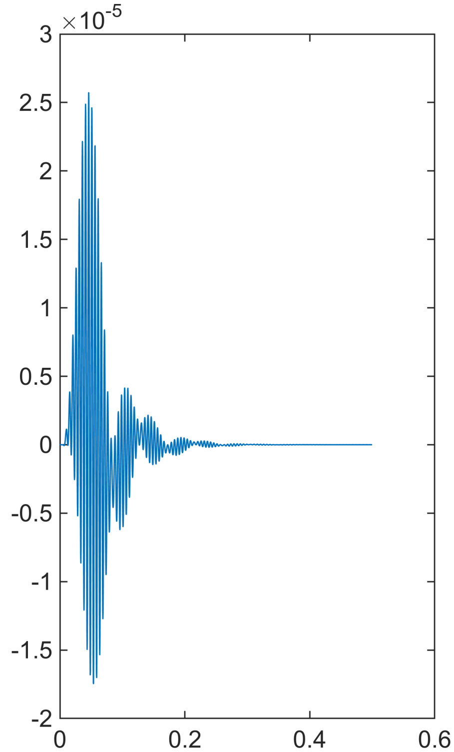
(d)

(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?

Create and plot the signal $y(t)$ and the output yo.

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





The outpupt yo fits the expected output.

(e)

(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 $m(t)$.

Assume $h_1(t) = m(t)cos(2\pi f_1 t)cos(2\pi f_1 t)$, $h_2(t) = m(t)cos(2\pi f_1 t)sin(2\pi f_1 t)$, $h_3(t) = m(t)cos(2\pi f_1 t)cos(2\pi f_2 t)$.

The continuous-time Fourier transform (CTFT) of a cosine of frequency f_1 is $C_1(j\omega) = \pi\delta(\omega - 2\pi f_1) + \pi\delta(\omega + 2\pi f_1)$.

Using $C_1(jw)$ and the multiplication property of the CTFT to obtain $H_1(j\omega)$.

$$
\begin{aligned}
F\{m(t)cos(2\pi f_1 t)\} &= \frac{1}{2}M(j(\omega-2\pi f_1)) + \frac{1}{2}M(j(\omega+2\pi f_1)) \\
F\{m(t)cos(2\pi f_1 t)cos(2\pi f_1 t)\} &= \frac{1}{2}(\frac{1}{2}M(j(\omega-4\pi f_1)) + \frac{1}{2}M(j\omega)) + \frac{1}{2}(\frac{1}{2}M(j\omega) + \frac{1}{2}M(j(\omega+4\pi f_1))) \\
H_1(j\omega) &= \frac{1}{4}M(j(\omega-4\pi f_1)) + \frac{1}{2}M(j\omega) + \frac{1}{4}M(j(\omega+4\pi f_1))
\end{aligned}
$$

The continuous-time Fourier transform (CTFT) of a cosine of frequency f_1 is

$S_1(j\omega) = \frac{\pi}{j}\delta(\omega - 2\pi f_1) - \frac{\pi}{j}\delta(\omega + 2\pi f_1)$.

Using $C_1(jw)$, $S_1(jw)$ and the multiplication property of the CTFT to obtain $H_2(j\omega)$.

$$
\begin{aligned}
F\{m(t)cos(2\pi f_1 t)\} &= \frac{1}{2}M(j(\omega-2\pi f_1)) + \frac{1}{2}M(j(\omega+2\pi f_1)) \\
F\{m(t)cos(2\pi f_1 t)sin(2\pi f_1 t)\} &= \frac{1}{2j}(\frac{1}{2}M(j(\omega-4\pi f_1)) + \frac{1}{2}M(j\omega)) - \frac{1}{2j}(\frac{1}{2}M(j\omega) + \frac{1}{2}M(j(\omega+4\pi f_1))) \\
H_2(j\omega) &= \frac{1}{4j}M(j(\omega-4\pi f_1)) - \frac{1}{4j}M(j(\omega+4\pi f_1))
\end{aligned}
$$

The continuous-time Fourier transform (CTFT) of a cosine of frequency f_2 is

$C_2(j\omega) = \pi\delta(\omega - 2\pi f_1) + \pi\delta(\omega + 2\pi f_1)$.

Using $C_1(jw)$, $C_2(jw)$ and the multiplication property of the CTFT to obtain $H_3(j\omega)$.

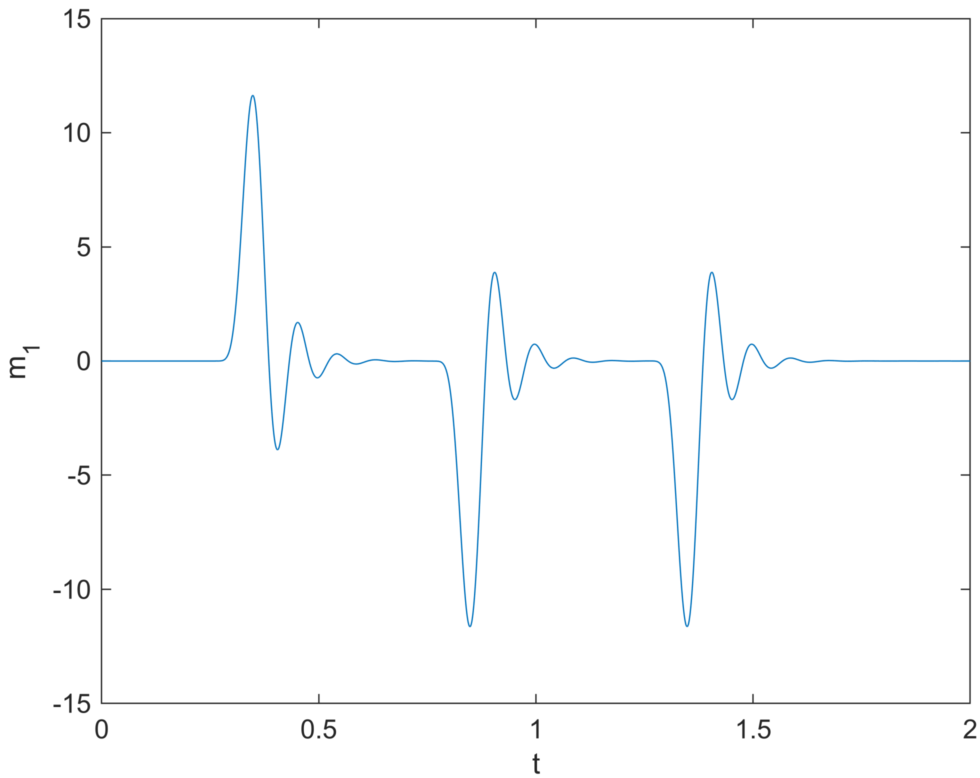
$$
\begin{aligned}
F\{m(t)cos(2\pi f_1 t)\} &= \frac{1}{2}M(j(\omega-2\pi f_1)) + \frac{1}{2}M(j(\omega+2\pi f_1)) \\
F\{m(t)cos(2\pi f_1 t)cos(2\pi f_2 t)\} &= \frac{1}{2}(\frac{1}{2}M(j(\omega-2\pi f_1-2\pi f_2)) + \frac{1}{2}M(j(\omega+2\pi f_1-2\pi f_2))) + \frac{1}{2}(\frac{1}{2}M(j(\omega-2\pi f_1+2\pi f_2)) + \frac{1}{2}M(j \\
H_3(j\omega) &= \frac{1}{4}M(j(\omega-2\pi f_1-2\pi f_2)) + \frac{1}{4}M(j(\omega+2\pi f_1-2\pi f_2)) + \frac{1}{4}M(j(\omega-2\pi f_1+2\pi f_2)) + \frac{1}{4}M(j(\omega+
\end{aligned}
$$

(f)

(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.

Decode $m_1(t)$ from $x(t)$, and plot them against t.

```
x1 = x .* cos(2*pi*f1*t(1:length(x)));
m1 = lsim(bf,af,x1,t(1:length(x1)));
figure;
plot(t(1:length(m1)),m1);xlabel('t');ylabel('m_1');
```



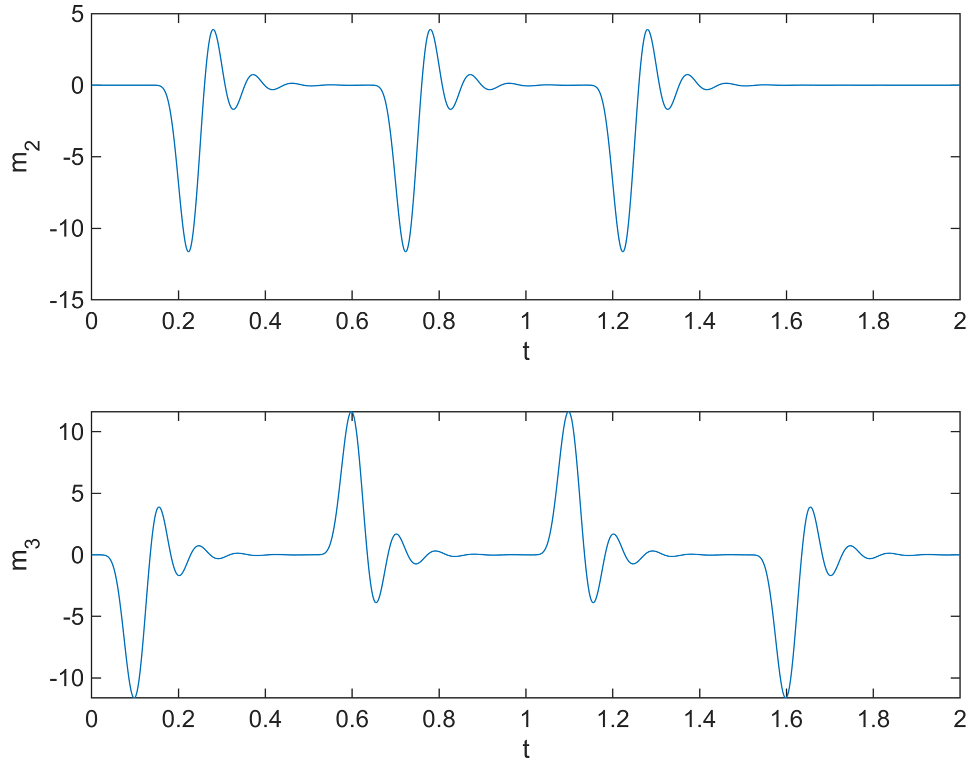
According to Morse code, m_1 means letter D.

(g)

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

Decode $m_2(t)$ and $m_3(t)$ from $x(t)$, and plot them against t.

```
x2 = x .* sin(2*pi*f2*t(1:length(x)));
m2 = lsim(bf,af,x2,t(1:length(x2)));
x3 = x .* sin(2*pi*f1*t(1:length(x)));
m3 = lsim(bf,af,x3,t(1:length(x3)));
figure;
subplot(2,1,1);
plot(t(1:length(m2)),m2);xlabel('t');ylabel('m_2');
subplot(2,1,2);
plot(t(1:length(m3)),m3);xlabel('t');ylabel('m_3');
```



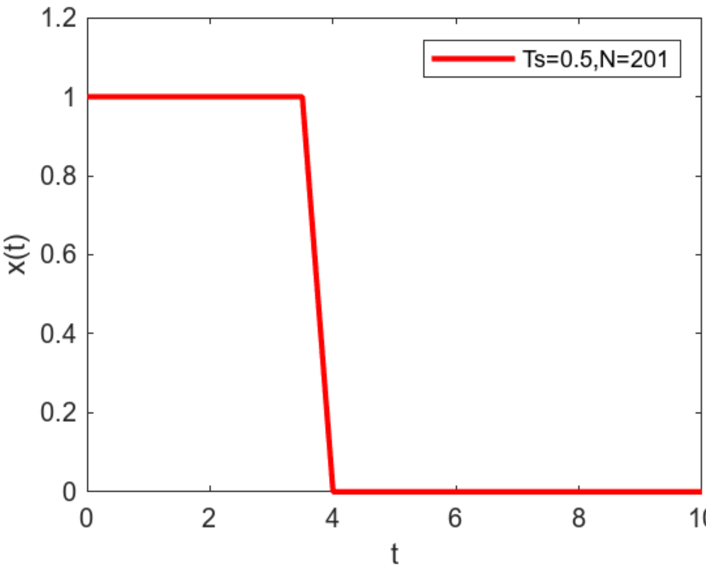
According to Morse code, m_2 means letter H.

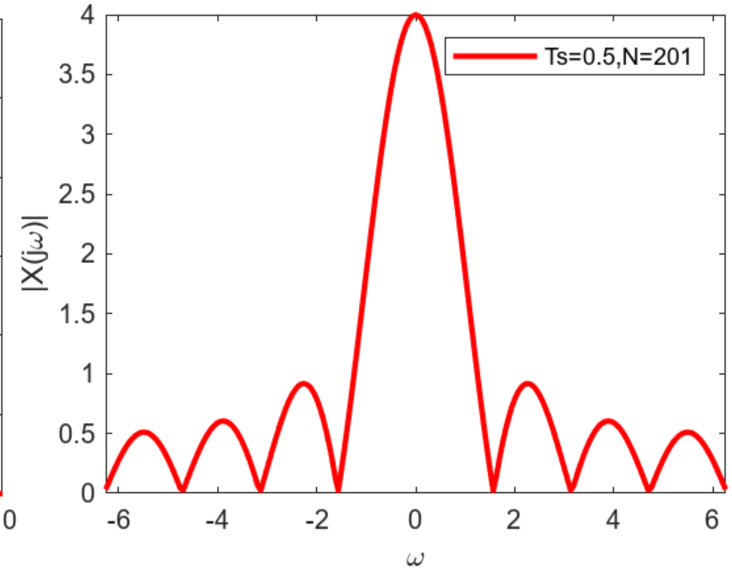
According to Morse code, m_3 means letter P.

Expeience

Through Matlab work, I have a preliminary understanding of Matlab simulation tools in the signal and system of the application of the discipline. I'm more famililar with the usage of approximation method of CTFT to solve problem. And I have a better understanding of the CTFT and the differences between approximation method and analytical method. I learn how to use MATLAB to compute numerical approximations to the CTFT integral, and explore amplitude modulation of Morse code messages.

欧阳安男（12211831）：

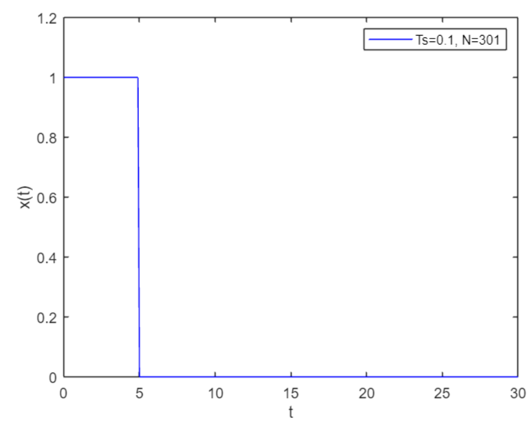


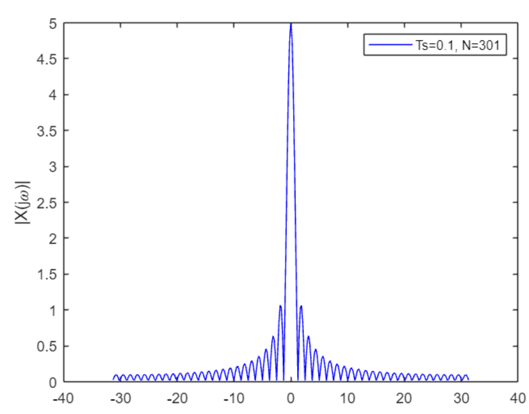


涂峻绫（12213010）：

untitled.mlx *

```
Ts = 0.1;
N = 301;
t = (0:N-1)*Ts;
x = [ones(1,5/Ts),zeros(1,N-5/Ts)];
figure;
plot(t,x,'b')
xlabel('t');ylabel('x(t)');
axis([0 30 0 1.2]);
legend('Ts=0.1, N=301');
X = fftshift(Ts*fft(x));
wl = -(N-1)*pi/N/Ts;
wu = (N-1)*pi/N/Ts;
step = 2*pi/N/Ts;
figure;
plot(wl:step:wu,abs(X),'b');
xlabel('\omega');ylabel('|X(j\omega)|');
legend('Ts=0.1, N=301');
```





Score

涂峻绫（100） 欧阳安男（100）