LAB3 MATLAB Programming

李志颖 12210309 夏琮杰 12210520

目录

Introduction

In this lab assignment, we learned about：

1.Usage and scenarios of fft() and ifft() functions in MATLAB

2.The concept of algorithmic time complexity

3.Try to call the above functions to analyse the signal in the frequency domain.

4.Understand the effect of filters on signals

5.Compare the time impact of using different algorithmic strategies to deal with real-world problems.

Results and Analysis

3.8

This exercise demonstrates the effect of first-order recursive discrete-time fi
signals. You will examine the frequency responses of two different system
struct a periodic signal to use as input for these systems. This exercise a
familiar with using **fft** and **ifft** to compute the DTFS of a periodic signa
Tutorial 3.1. In addition, it is also assumed you are proficient with the **fi**
commands described in Tutorials 2.2 and 3.2. Several parts of this exercis
generate vectors which should be purely real, but have very small imagina
roundoff errors. Use **real** to remove these residual imaginary parts from t

This exercise focuses on two causal LTI systems described by first-ord
ference equations:

$$\begin{aligned}\text{System 1:} \quad y[n] - 0.8y[n-1] &= x[n], \\ \text{System 2:} \quad y[n] + 0.8y[n-1] &= x[n].\end{aligned}$$

The input signal $x[n]$ will be the periodic signal with period $N = 20$ describ
coefficients

$$a_k = \begin{cases} 3/4\,, & k = \pm 1\,, \\ -1/2\,, & k = \pm 9\,, \\ 0\,, & \text{otherwise}\,. \end{cases}$$

(a)

(a). Define vectors **a1** and **b1** for the difference equation describing System
specified by **filter** and **freqz**. Similarly, define **a2** and **b2** to describ

In Matlab, the coefficients at the output and input of the system are recorded using ai, bi, respectively.For the system in this question , we have a1=[1 -0.8],a2=[1 0.8],b1=b2=1

```
%System 1
a1=[1 -0.8];
b1=1;
y1=filter(b1,a1,x);
```

```
%System 2
a2=[1 0.8];
b2=1;
y2=filter(b2,a2,x);
```

(b)

(b). Use **freqz** to evaluate the frequency responses of Systems 1 and 2 between 0 and 2π. Note that you will have to use the **'whole'** option do this. Use **plot** and **abs** to generate appropriately labeled graphs of of the frequency response for both systems. Based on the frequency specify whether each system is a highpass, lowpass, or bandpass filter

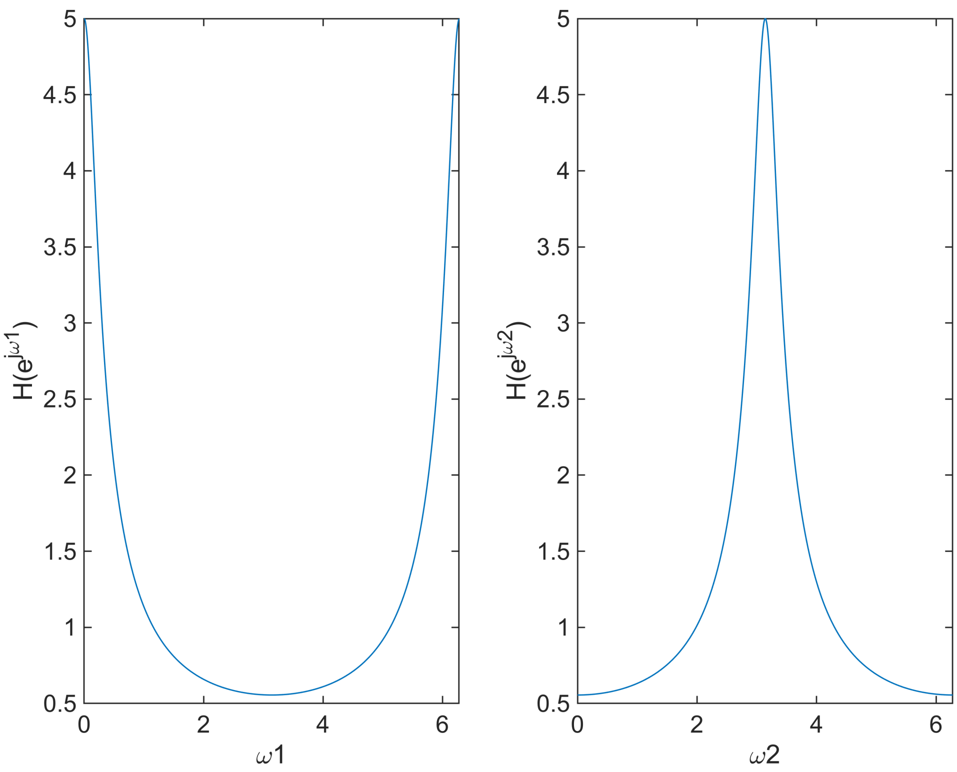
n=1024

```
n=1024
```

```
n = 1024
```

```
%System 1
a1=[1 -0.8];
b1=1;
[H1, Omega1]=freqz(b1,a1,n,'whole');
subplot(1,2,1);
plot(Omega1,abs(H1));
xlabel('\omega1');ylabel('H(e^{j\omega1})');


%System 2
a2=[1 0.8];
b2=1;
[H2, Omega2]=freqz(b2,a2,n,'whole');
subplot(1,2,2);
plot(Omega2,abs(H2));
xlabel('\omega2');ylabel('H(e^{j\omega2})');
```



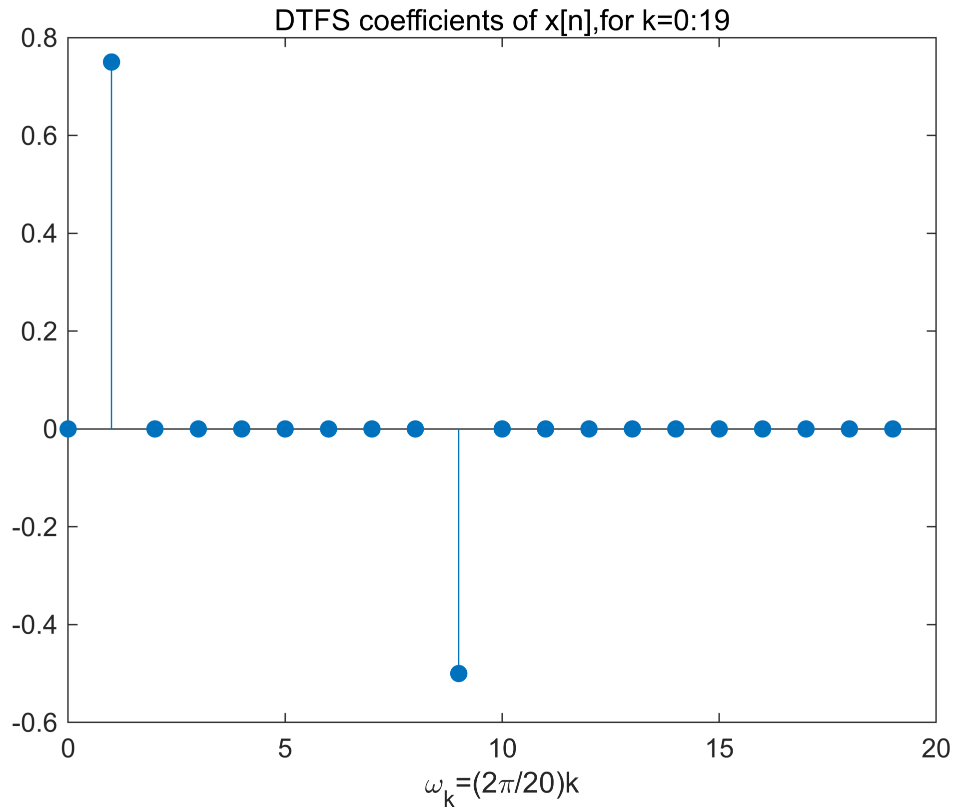
From the image, system 1 is a low pass filter and system 2 is a high pass filter.

(C)

(c). Use Eq. (3.10) to define the vector **a_x** to be the DTFS coefficients of
19. Generate a plot of the DTFS coefficients using **stem** where the x
with frequency $\omega_k = (2\pi/20)k$. Compare this plot with the frequency
generated in Part (b), and for each system state which frequency com
amplified and which will be attenuated when $x[n]$ is the input to the

Create the vector a_x and graph it

```
clear; clc; close all;
a_x = zeros(1, 20);
a_x(2) = 3 /4;
a_x(10) = -0.5;
stem(0:19, a_x, 'filled');
title('DTFS coefficients of x[n],for k=0:19');
xlabel('\omega_k=(2\pi/20)k');
```



System 1 will amplify component k=1 and attenuate component k=9. System 2 will do the opposite..

(d)

(d). Define **x_20** to be one period of $x[n]$ for $0 \leq n \leq 19$ using **ifft** with in Tutorial 3.1. Use **x_20** to create **x**, consisting of 6 periods of $x[n]$ fo Also define **n** to be this range of sample indices, and generate a plo interval using **stem**.

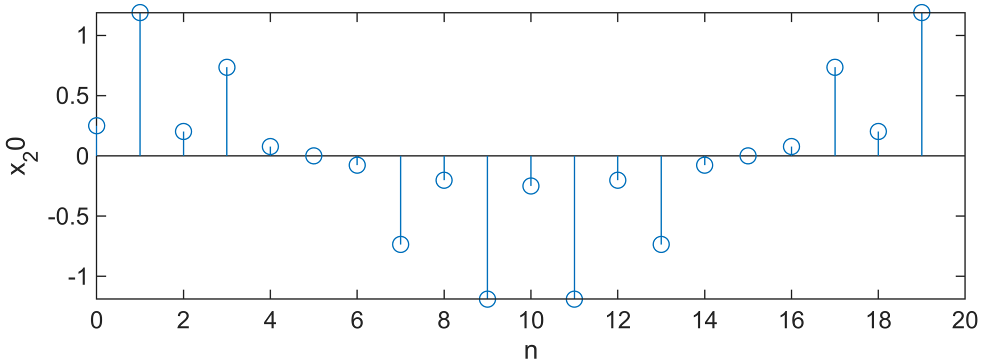
Use the ifft function to represent x_20, then use the repmat function to record the value of x over multiple cycles for the graph.

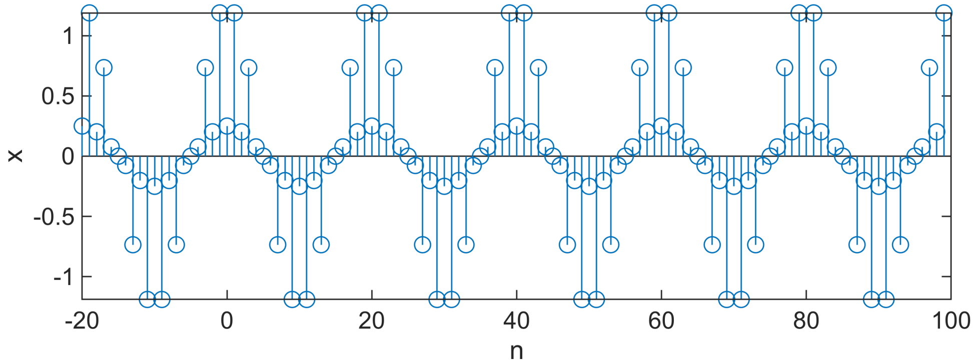
```
x_20 = ifft(a_x) * length(a_x);
n = -20: 99;

%plot;
subplot(2,1,1);
stem(0:19,real(x_20));
xlabel('n');ylabel('x_20');

x=real(repmat(x_20,1,6));
subplot(2,1,2);
stem(n,x);
```

```
xlabel('n');
ylabel('x');
```





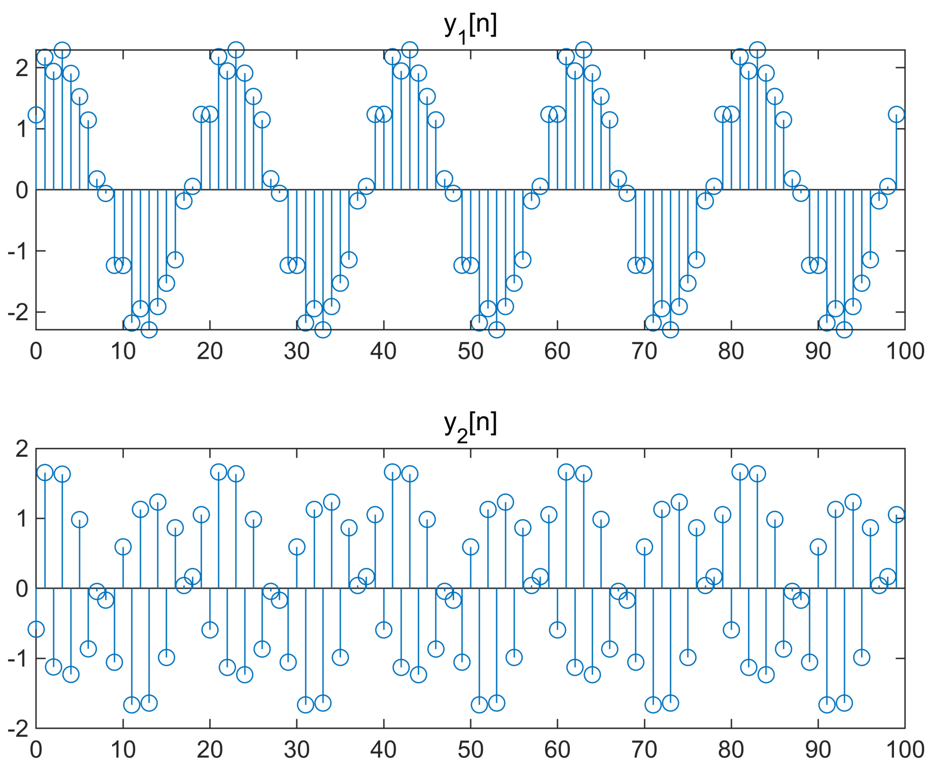
(e)

(e). Use **filter** to compute **y1** and **y2**, the outputs of Systems 1 and 2 w input. Plot both outputs for $0 \leq n \leq 99$ using **stem**. Based on th which output contains more high frequency energy and which has mor energy. Do the plots confirm your answers in Part (c)?

Use the filter function to obtain the output signals y1, y2 and graph them.

```
clear; clc; close all;
a_x = zeros(1, 20);
a_x(2) = 3 /4;
a_x(10) = -0.5;
x_20 = ifft(a_x) * length(a_x);
x=real(repmat(x_20,1,6));
%System 1
a1=[1 -0.8];
b1=1;
y1=filter(b1,a1,x);
```

```
%System 2
a2=[1 0.8];
b2=1;
y2=filter(b2,a2,x);
n=-20:99;
subplot(2, 1, 1);
stem(n, y1);
xlim([0, 100]);
title('y_1[n]');
subplot(2, 1, 2);
stem(n, y2);
xlim([0, 100]);
title('y_2[n]');
```



y1 has more low frequency energy. y2 has more high frequency energy. It confirmed the answers in part c.

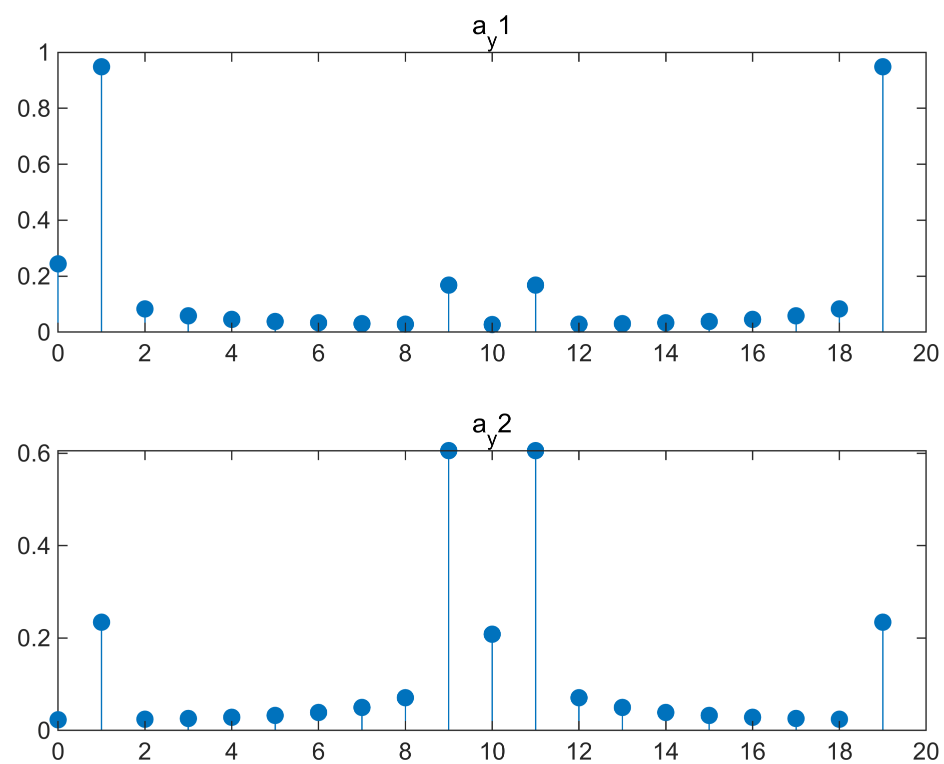
(f)

(f). Define `y1_20` and `y2_20` to be the segments of `y1` and `y2` correspond $y_2[n]$ for $0 \leq n \leq 19$. Use these vectors and `fft` to compute `a_y1` and coefficients of `y1` and `y2`. Use `stem` and `abs` to generate plots of th the DTFS coefficients for both sequences. Do these plots agree with Part (e)?

According to the question, take a one period y signal and obtain its Fourier series coefficients by fft function and make a graph.

```
y1_20 = y1(1:20);
y2_20 = y2(1:20);
a_y1 = fft(y1_20) / length(y1_20);
a_y2 = fft(y2_20) / length(y2_20);

subplot(2, 1, 1);
stem(0:19, abs(a_y1), 'filled');
title('a_y1');
subplot(2, 1, 2);
stem(0:19, abs(a_y2), 'filled');
title('a_y2');
```



Comparison with the Fourier level coefficient plots of the input signals reveals that the high frequency component is suppressed after system 1 and the low frequency component is suppressed after system 2.The plots agreed with the answers in part e.

3.10

The FFT Algorithm for Computing the DTFS

The DTFS for a periodic discrete-time signal with fundamental period
Eq. (3.2).

(a)

(a). Argue that computing each coefficient a_k requires $N+1$ complex mu
$N-1$ complex additions. Assume that the functions $e^{-jk(2\pi/N)n}$ do
computation. Using this result, how many operations are required
DTFS for a signal with fundamental period N? Note that the numb
required is independent of the particular signal $x[n]$.

N+1 complex multiplications for computing a0 and aN/2;

2(N-1) complex multiplications for computing the remaining coefficients;

N-1 complex additions for computing all coefficients;

So, the total number of operations required to compute the DTFS is N+1+2(N-1)+N-1 = **4N-2.**

(b)

(b). If you have not already done the Advanced Problem in Exercise 3.5
so now. You will compare the amount of computation this algorithm

amount required by **fft**. You can measure the number of operations use
to implement Eq. (3.2) by using the internal **flops** (floating point opera
additions and multiplications) counter of MATLAB as follows:

```
>> x = (0.9).^[0:N-1]; % create one period of x[n]
>> flops(0);  % set MATLAB's internal computation counter to 0
>> X = dtfs(x,0); % Store the DTFS of x[n] in X
>> c = flops;  % Store the number of operations in c
```

Find c for computing X using dtfs for $N = 8, 32, 64, 128$, and 256. Save t
in the vector dtfscomps.

Since the new version of matlab removes the flop function, we use tic,toc to computing X using dtfs and fft for N=8,32,64,128,and 256. save them in the dtfscomps and fftcomps

```
clear; clc; close all;
N = [8, 32, 64, 128, 256];
dtfscomps = zeros(1, length(N));
fftcomps = zeros(1, length(N));
for i = 1:length(N)
    x = (0.9).^(0:N(i)-1);
    tic;
    dtfs(x,0);
    c1 = toc;
    dtfscomps(i) = c1;
    tic;
    fft(x) / 5;
    c2=toc;
    fftcomps(i) = c2;
end
dtfscomps
```

```
dtfscomps = 1×5
10^-3 ×
    0.2425    0.0507    0.0832    0.0412    0.1117
```

(c)

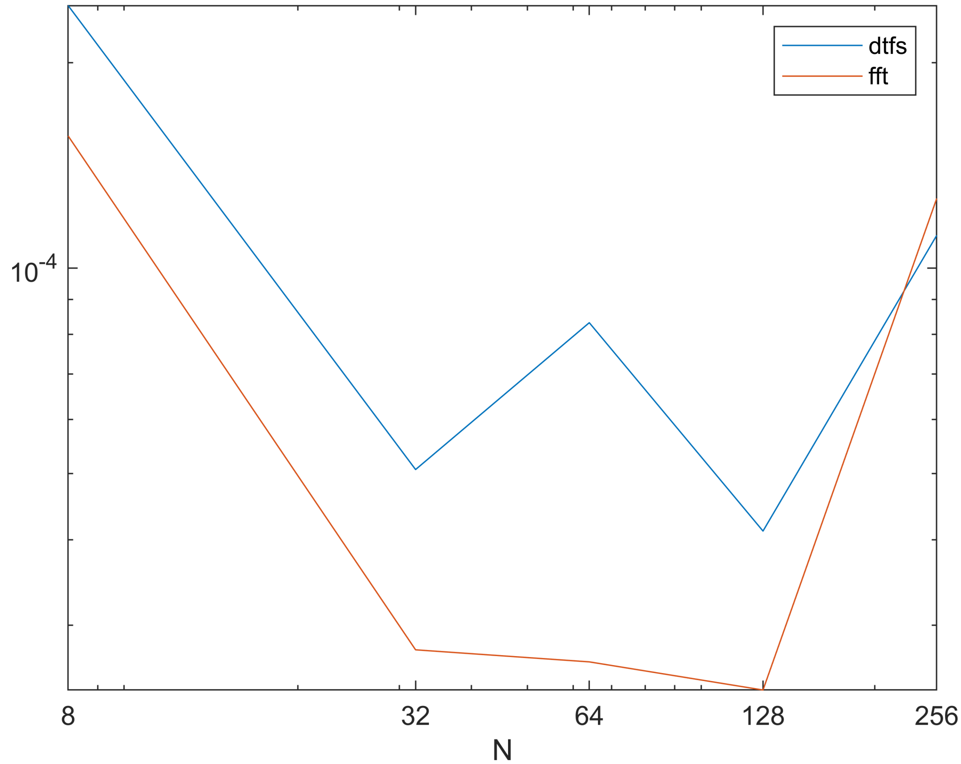
(c). Now, compute the DTFS coefficients of $x[n]$ for $N = 8, 32, 64, 128$
fft as shown in Tutorial 3.1. Use **flops** to find the number of ope
each value of N and store these values in the vector **fftcomps**. Plot
fftcomps versus N using **loglog**. How does the number of operati
fft compare to that required by **dtfs**, particularly for large values o

plot

```
fftcomps
```

```
fftcomps = 1×5
10^-3 ×
    0.1562    0.0276    0.0265    0.0241    0.1265
```

```
figure;
loglog(N,dtfscomps);
xticks(N);
hold on;
loglog(N,fftcomps);
legend('dtfs','fft');
xlabel('N');
```



(d)

(d). What is the fundamental period, N_y, of $y[n]$? Argue that directly im
periodic convolution according to Eq. (3.11) requires $\mathcal{O}(N^2)$ operat
and multiplications). Remember that computing one period of $y[n]$
characterize the entire signal.

The fundamental period , N_y, of y[n] is N;

$$y[n+N] = \sum_{r=0}^{N-1} x[r]h[n+N-r] = \sum_{r=0}^{N-1} x[r]h[n-r] = y[n]$$

n is the peiod of y[n], and the fundamental period of h[n].

(e)

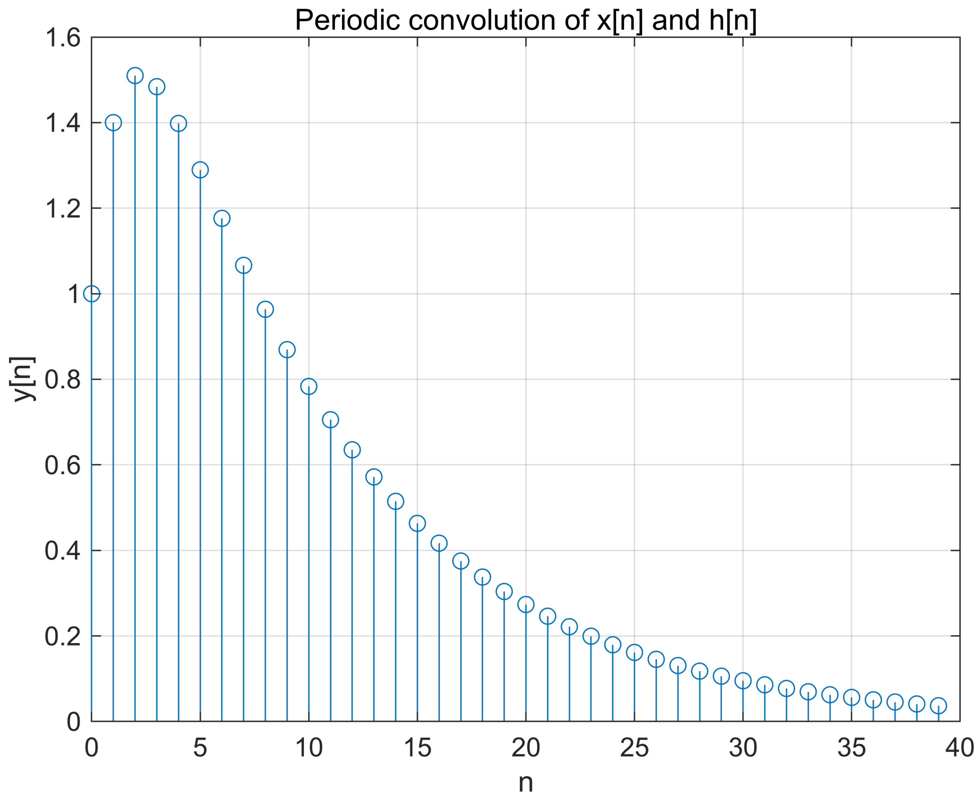
(e). Assume both $x[n]$ and $h[n]$ have fundamental period $N = 40$, and are
$(0.9)^n$ and $h[n] = (0.5)^n$ over the interval $0 \le n \le N-1$. Compu
convolution of $x[n]$ with $h[n]$ and plot $y[n]$ for $0 \le n \le N_y - 1$. Store
operations, given by **flops**, required to implement the convolution in **f4**
to set **flops(0)** after creating **x** and **h**, the vectors representing $x[n]$
To implement the periodic convolution, first store $x[n]$ and $h[n]$ ov
$0 \le n \le N-1$ in the row vectors **x** and **h**, respectively. Set **flops(0**
conv([x x],h). The periodic convolution can be extracted from a
signal.

```
clear; clc; close all;
N = 40;
x = (0.9).^(0:N-1);
h = (0.5).^(0:N-1);
n = 0+0:(2*N-1)+(N-1);
tic;
y = conv([x x], h);
f40c = toc;
f40c
```

```
f40c = 0.0015
```

```
stem(n(1:N), y(1:N));
xlabel('n'); ylabel('y[n]');
title('Periodic convolution of x[n] and h[n]');
```

```
grid on;
```



(f)

(f). Repeat Part (e) for $N = 80$, again plotting a period of $y[n]$ and stor
of operations in `f80c`.

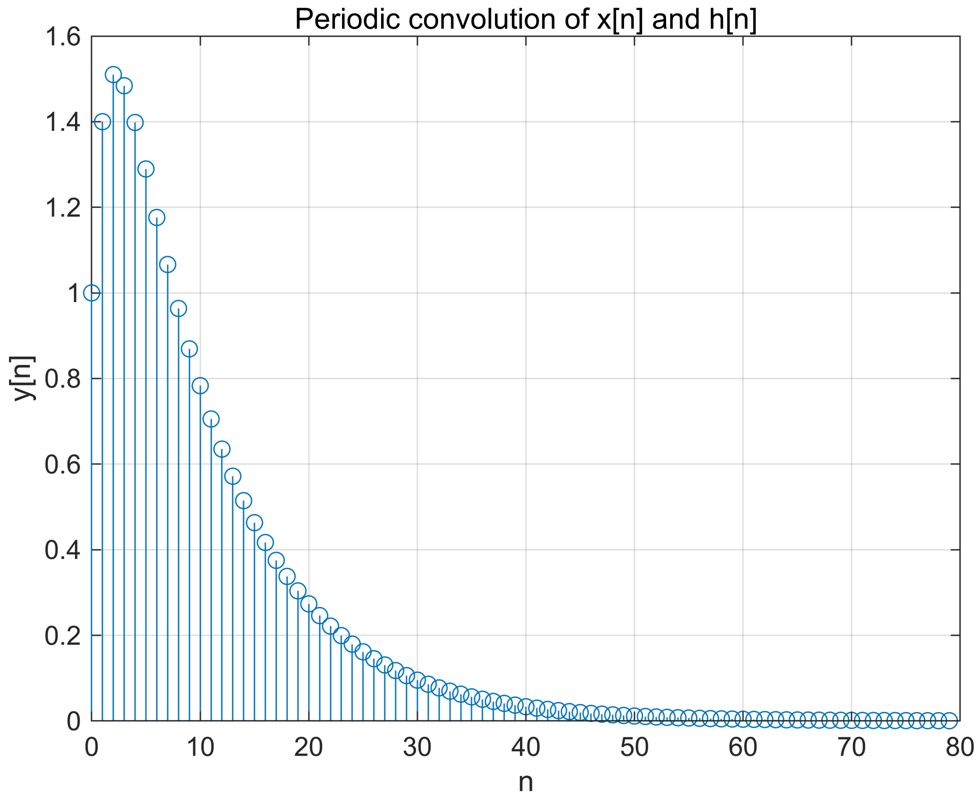
same as (e)

```
clear; clc; close all;
N = 80;
x = (0.9).^(0:N-1);
h = (0.5).^(0:N-1);
n = 0+0:(2*N-1)+(N-1);
tic;
y = conv([x x], h);
f80c = toc;
f80c
```

```
f80c = 0.0018
```

```
stem(n(1:N), y(1:N));
xlabel('n'); ylabel('y[n]');
```

```
title('Periodic convolution of x[n] and h[n]');
grid on;
```



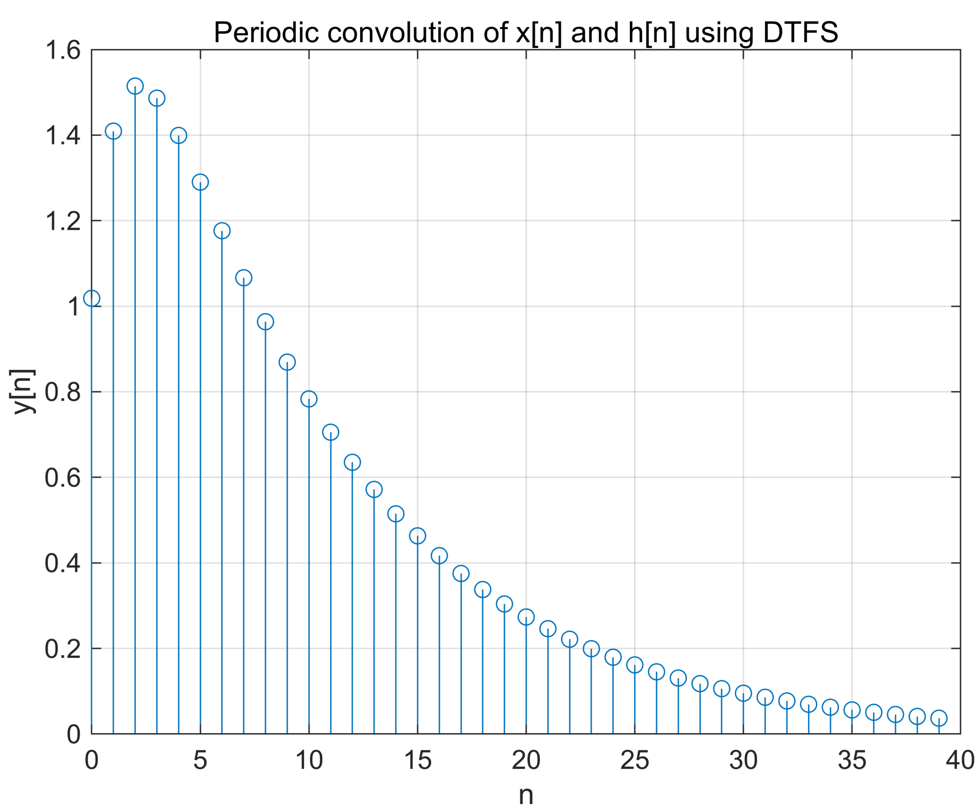
(g)

(g). Assume $x[n]$ and $h[n]$ are defined as in Part (e). Use the periodic conv of the DTFS to implement the periodic convolution[2]. Namely, com coefficients of both $x[n]$ and $h[n]$ using **fft** as described in Tutorial 3. periodic convolution property to form the DTFS coefficients of $y[n]$. Fi $y[n]$ from the DTFS coefficients using **ifft**. The **ifft** algorithm is to the FFT, and also requires $\mathcal{O}(N \log N)$ operations for a signal w period N. To check the validity of your implementation, plot $y[n]$ for and compare this signal to that computed in Part (e). Remember, Tutorial 3.1, the signal $y[n]$ might have a small imaginary component round-off errors. Store the total number of operations required to **f40f**, and remember to set **flops(0)** after creating the representat $h[n]$ in MATLAB.

```
clear; clc; close all;
N = 40;
x = (0.9).^(0:N-1);
h = (0.5).^(0:N-1);
n = 0+0:(2*N-1)+(N-1);
tic;
X = fft(x) / N;
H = fft(h) / N;
Y = N.*X .* H;
y = ifft(Y) * N;
f40f = toc;
f40f
```

```
f40f = 0.0043
```

```
stem(n(1:N),y(1:N));
xlabel('n'); ylabel('y[n]');
title('Periodic convolution of x[n] and h[n] using DTFS');
grid on;
```



(h)

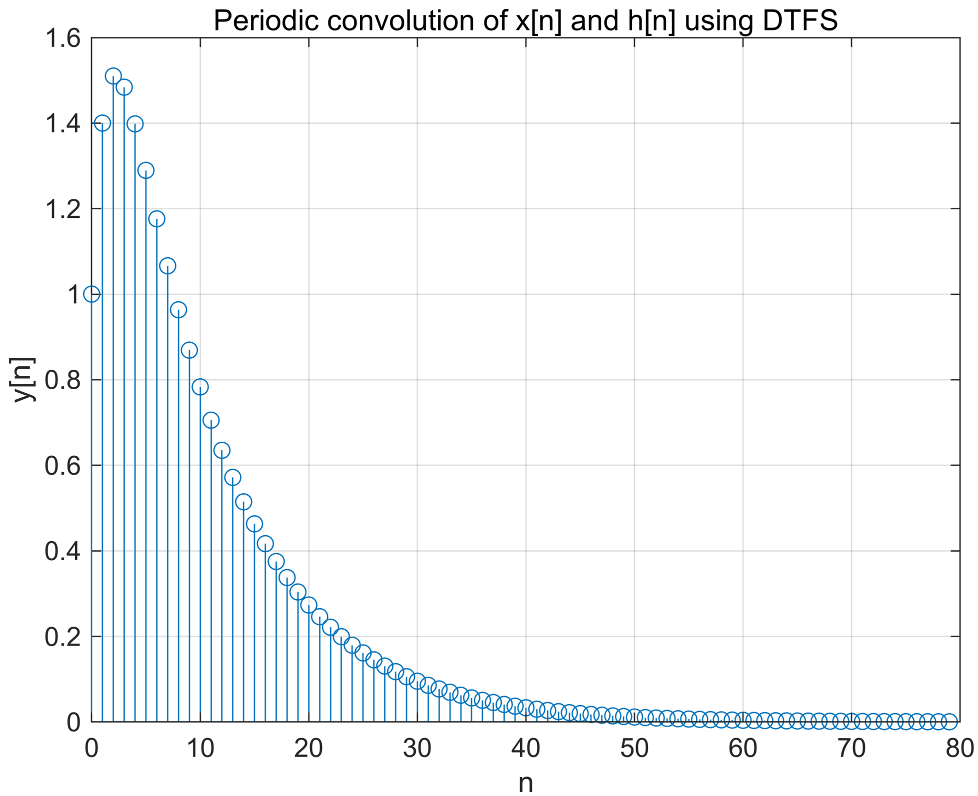
(h). Repeat Part (g) for $N = 80$, again plotting a period of $y[n]$ and stori
of operations in **f80f**. Again check the validity of your implementatio
$y[n]$ with that computed in Part (f).

same as (g)

```
clear; clc; close all;
N = 80;
x = (0.9).^(0:N-1);
h = (0.5).^(0:N-1);
n = 0+0:(2*N-1)+(N-1);
tic;
X = fft(x) / N;
H = fft(h) / N;
Y = N.*X .* H;
y = ifft(Y) * N;
f80f = toc;
f80f
```

```
f80f = 0.0047
```

```
stem(n(1:N),y(1:N));
xlabel('n'); ylabel('y[n]');
title('Periodic convolution of x[n] and h[n] using DTFS');
grid on;
```



(i)

(i). Compute the ratios of **f40c** to **f40f** and **f80c** to **f80f**. How do thes
for $N = 40$ and $N = 80$? Which method of convolution is more efficie
of N? Which method would you choose for $N > 80$? Justify your an

```
clear; clc; close all;
f40c=0.0015; f80c=0.0018;f40f=0.0043;f80f=0.0047;
ratio_40=f40c / f40f;
ratio_80=f80c / f80f;
ratio_40
```

```
ratio_40 = 0.3488
```

```
ratio_80
```

```
ratio_80 = 0.3830
```

these ratios increase as N increases, which means that the difference in efficiency between the two methods becomes more significant for larger values of N.

Therefore, the method of convolution that is more efficient for each value of N is the one that uses the periodic convolution property of the DTFS

This method would also be the preferred choice for N > 80

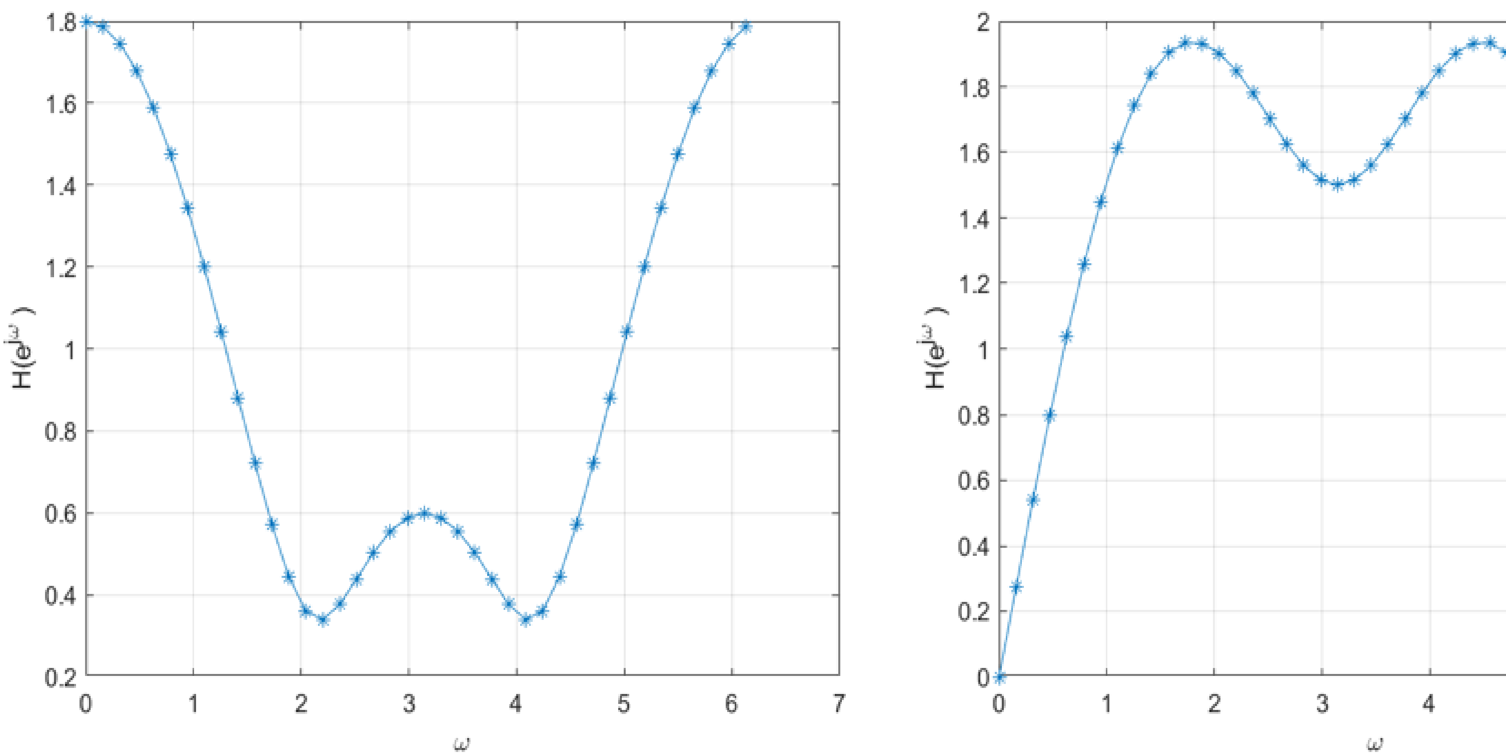
Function

```
function X = dtfs(x, k)
N = length(x);
if nargin < 2
    k = 0:N-1;
end
X = zeros(1, length(k));
for i = 1:length(k)
    n = 0:N-1;
    X(i) = sum(x .* exp(-1j * 2 * pi * k(i) * n / N)) / N;
end
end
```

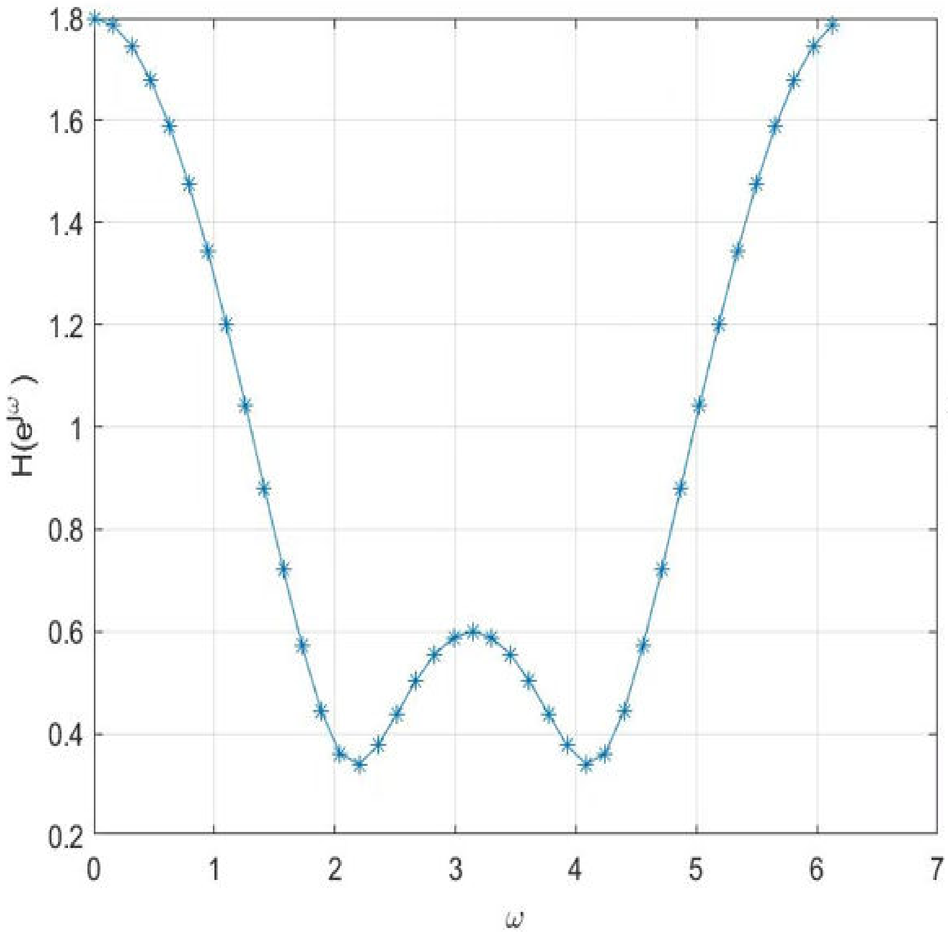
Experience

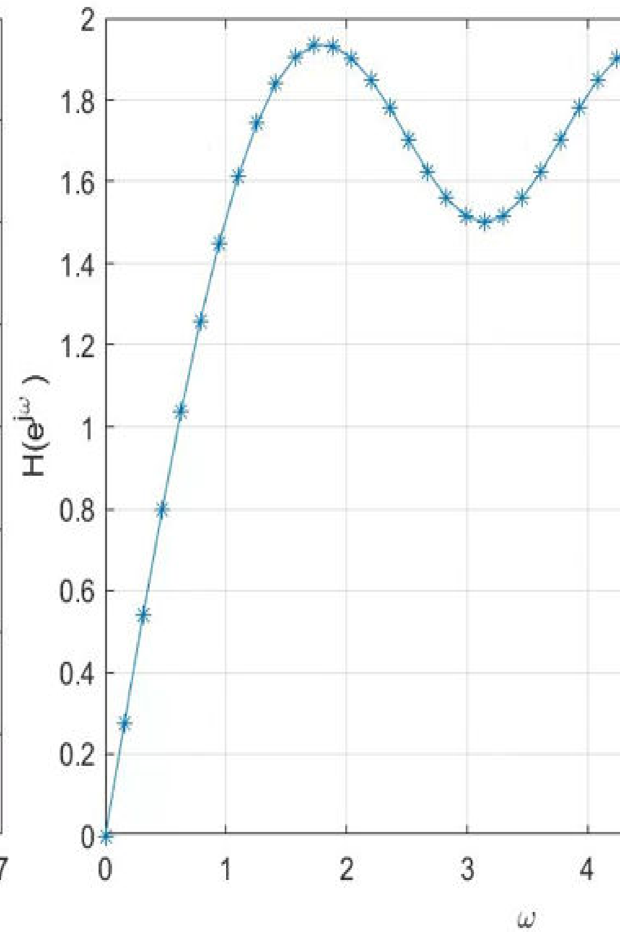
exercise figure

12210309 李志颖



12210520 夏琮杰





1.We have learnt about the use of the fft(),which can be used to decompose a signal: converting a signal into its Fourier series representation.

2.We have learnt about the use of the ifft() ,which performs the synthesis of the signal: it represents a series of signals corresponding to the Fourier series coefficients.

3.We learnt how filters work: signals of different frequencies are amplified or reduced in a specific way to achieve the effects of "high pass", "low pass" and "band pass".

4.When choosing an algorithm to solve a problem, it is important to consider the time complexity and other relevant parameters while satisfying the functionality to achieve the most efficient use of the code to complete the task.

5.We realize that different systems do have different filtering effects