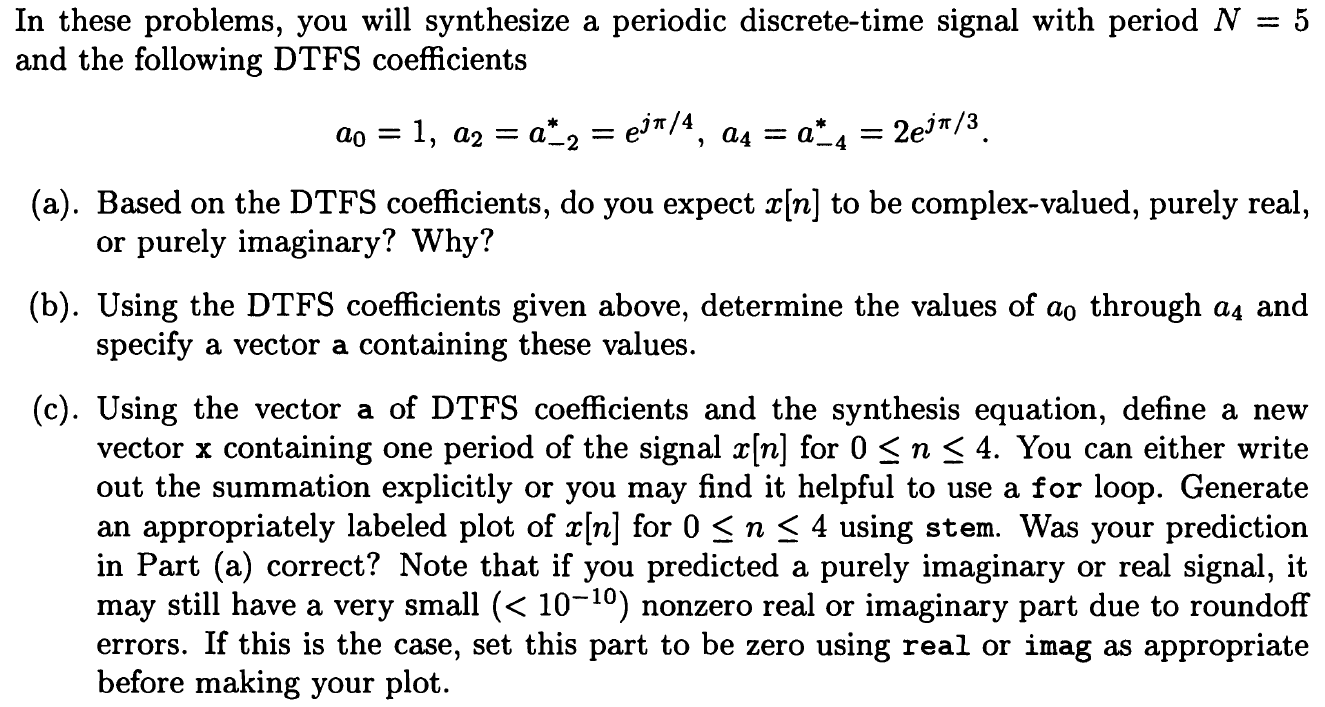
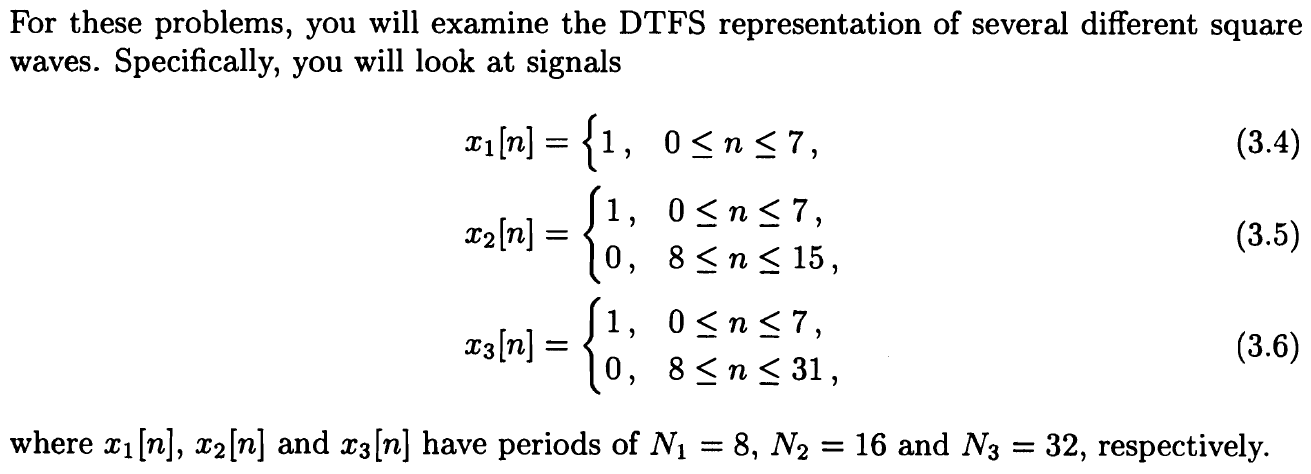
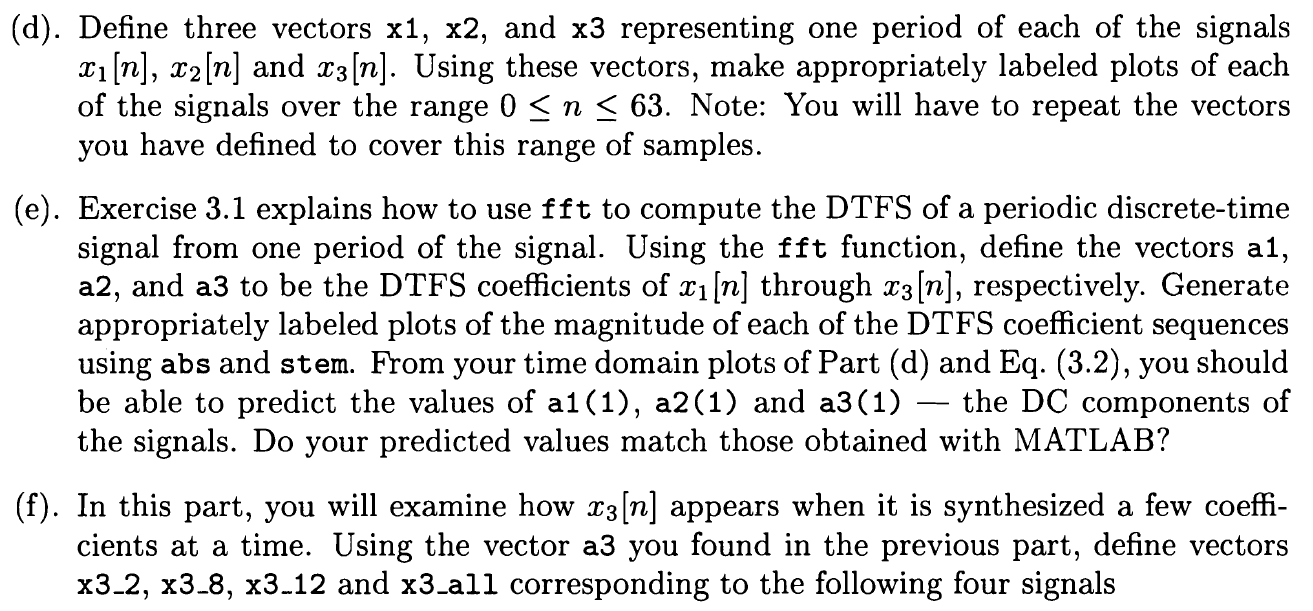
Name 1:李璇 SID 1:12010137 Name 2:张林燊SID 2:12010424

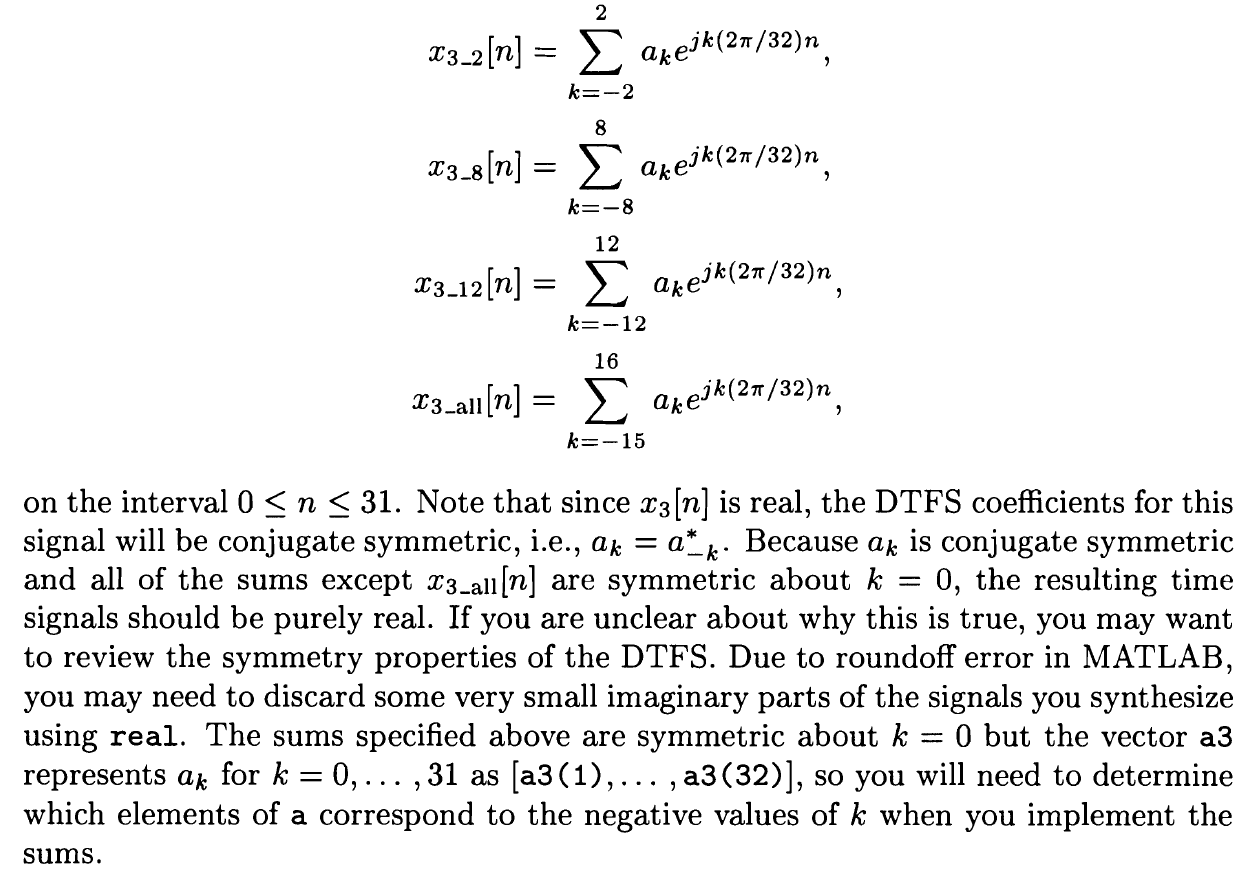
实验报告#3 (Lab#3)

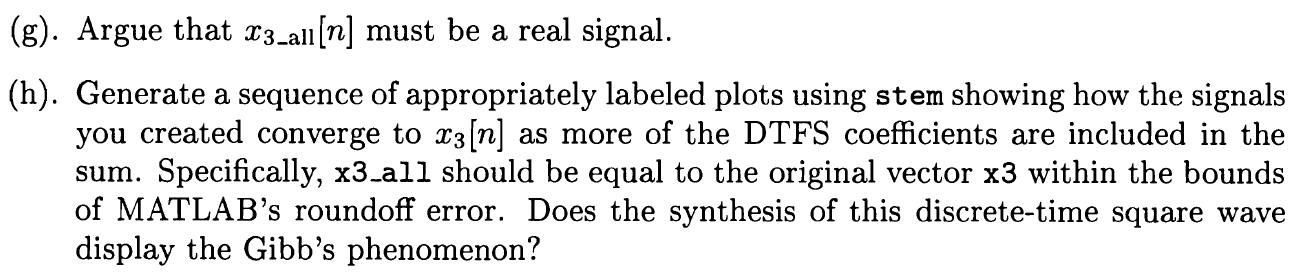
**3.5**

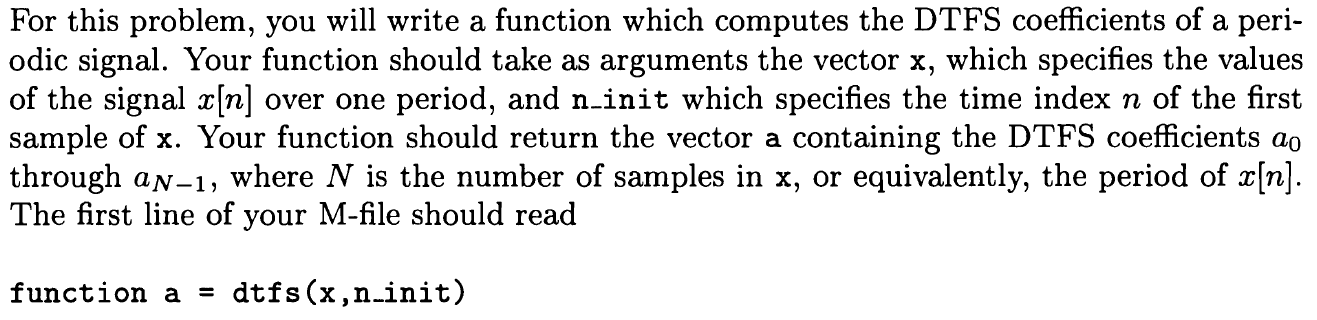






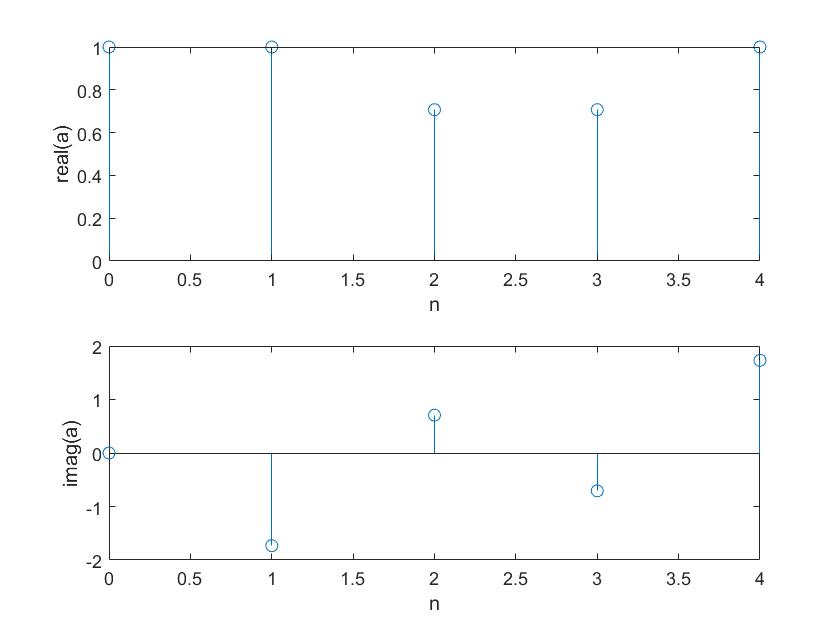






**Solution:**

1. The coefficient indicates that ak=a-k\*, then the terms with j are eliminated and only real terms are remained. So x[n] should be purely real.
2. Since N=5, then due to the periodic property, a-1=a4=2ejΠ/3, a-3=a2=ejΠ/4. Thus, a0=1, a1=a-1\*=2e-jΠ/3, a2=a-2\*=ejΠ/4, a3=a-3\*=e-jΠ/4, a4=a-4\*=2ejΠ/3.

****

**Matlab code:**

n=0:4;

a(1)=1;

a(2)=2\*exp(-1i\*pi\*(1/3));

a(3)=exp(1i\*pi\*(1/4));

a(4)=exp(-1i\*pi\*(1/4));

a(5)=2\*exp(1i\*pi\*(1/3));

subplot(2,1,1);

stem(n,real(a));

xlabel('n');

ylabel('real(a)');

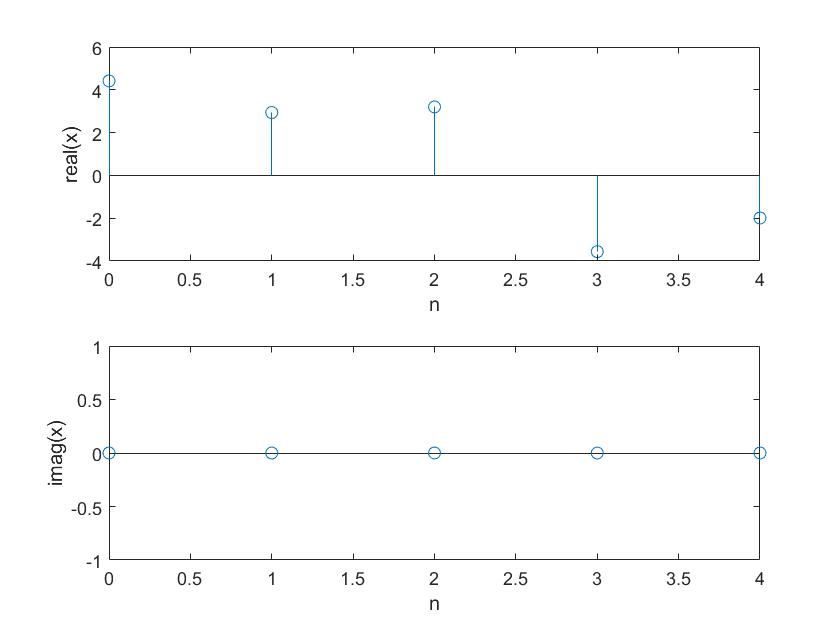
subplot(2,1,2);

stem(n,imag(a));

xlabel('n');

ylabel('imag(a)');

1. According to the graph, the imaginary part of x is zero, so x[n] is purely real, which indicates that the prediction in (1) is correct.

****

**Matlab code:**

n=0:4;

a(1)=1;

a(2)=2\*exp(-1i\*pi\*(1/3));

a(3)=exp(1i\*pi\*(1/4));

a(4)=exp(-1i\*pi\*(1/4));

a(5)=2\*exp(1i\*pi\*(1/3));

x=5\*ifft(a);

subplot(2,1,1);

stem(n,real(x));

xlabel('n');

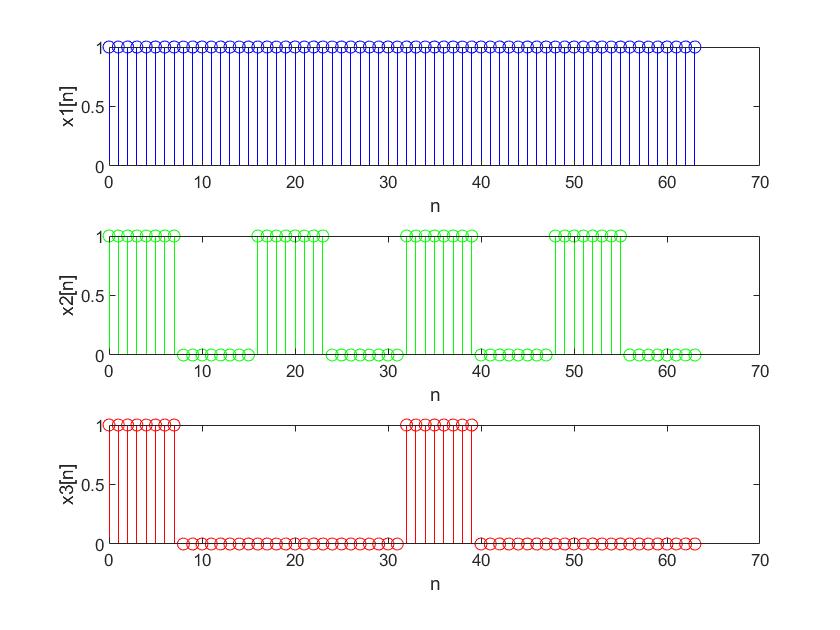
ylabel('real(x)');

subplot(2,1,2);

stem(n,imag(x));

xlabel('n');

ylabel('imag(x)');



**Matlab code:**

n1=0:7;

n2=0:15;

n3=0:31;

x1=ones(1,8);

x2=[ones(1,8),zeros(1,8)];

x3=[ones(1,8),zeros(1,24)];

subplot(3,1,1);

for k=0:7

stem(n1+8\*k,x1,'b');

hold on;

k+1;

end

xlabel('n');

ylabel('x1[n]');

subplot(3,1,2);

for m=0:3

stem(n2+16\*m,x2,'g');

hold on;

m+1;

end

xlabel('n');

ylabel('x2[n]');

subplot(3,1,3);

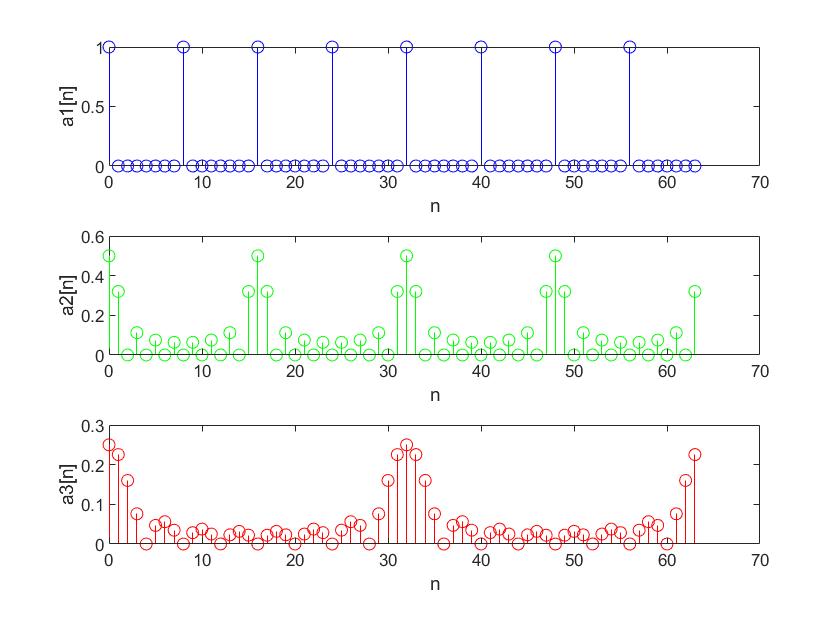
stem(n3,x3,'r');

hold on;

stem(n3+32,x3,'r');

xlabel('n');

ylabel('x3[n]');



**Matlab code:**

n1=0:7;

n2=0:15;

n3=0:31;

x1=ones(1,8);

x2=[ones(1,8),zeros(1,8)];

x3=[ones(1,8),zeros(1,24)];

a1=(1/8)\*fft(x1);

a2=(1/16)\*fft(x2);

a3=(1/32)\*fft(x3);

subplot(3,1,1);

for k=0:7

stem(n1+8\*k,abs(a1),'b');

hold on;

k+1;

end

xlabel('n');

ylabel('a1[n]');

subplot(3,1,2);

for m=0:3

stem(n2+16\*m,abs(a2),'g');

hold on;

m+1;

end

xlabel('n');

ylabel('a2[n]');

subplot(3,1,3);

stem(n3,abs(a3),'r');

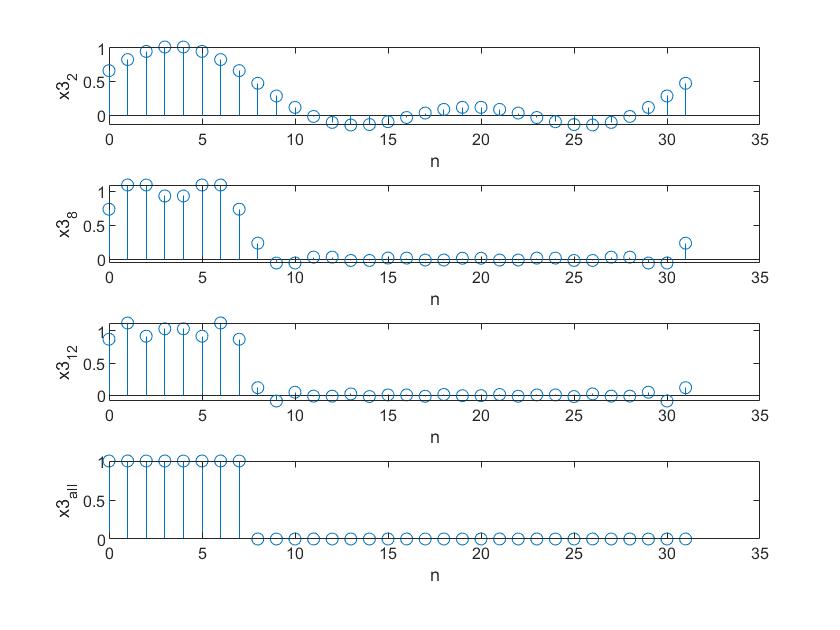
hold on;

stem(n3+32,abs(a3),'r');

xlabel('n');

ylabel('a3[n]');

From Eq.(3.2), it could be predicted that a1(1)=1, a2(1)=1/2, a3(1)=1/4.



**Matlab code:**

n0=0:31;

x3=[ones(1,8),zeros(1,24)];

a3=(1/32)\*fft(x3);

x32=zeros(1,32);

x38=zeros(1,32);

x312=zeros(1,32);

x3all=zeros(1,32);

for n=0:31

for k=-2:-1

x32(n+1)=x32(n+1)+conj(a3(-k+1)).\*exp(1i\*k\*n\*(2\*pi/32));

end

for k=0:2

x32(n+1)=x32(n+1)+a3(k+1).\*exp(1i\*k\*n\*(2\*pi/32));

end

end

for n=0:31

for k=-8:-1

x38(n+1)=x38(n+1)+conj(a3(-k+1)).\*exp(1i\*k\*n\*(2\*pi/32));

end

for k=0:8

x38(n+1)=x38(n+1)+a3(k+1).\*exp(1i\*k\*n\*(2\*pi/32));

end

end

for n=0:31

for k=-12:-1

x312(n+1)=x312(n+1)+conj(a3(-k+1)).\*exp(1i\*k\*n\*(2\*pi/32));

end

for k=0:12

x312(n+1)=x312(n+1)+a3(k+1).\*exp(1i\*k\*n\*(2\*pi/32));

end

end

for n=0:31

for k=-15:-1

x3all(n+1)=x3all(n+1)+conj(a3(-k+1))\*exp(1i\*k\*n\*(2\*pi/32));

end

for k=0:16

x3all(n+1)=x3all(n+1)+a3(k+1)\*exp(1i\*k\*n\*(2\*pi/32));

end

end

subplot(4,1,1);

stem(n0,real(x32));

xlabel('n');

ylabel('x3\_2');

subplot(4,1,2);

stem(n0,real(x38));

xlabel('n');

ylabel('x3\_8');

subplot(4,1,3);

stem(n0,real(x312));

xlabel('n');

ylabel('x3\_1\_2');

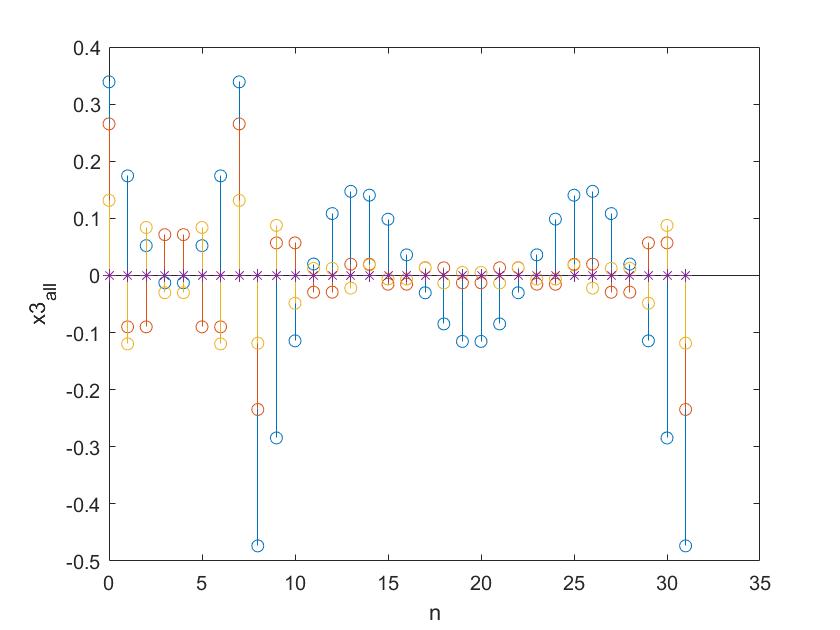
subplot(4,1,4);

stem(n0,real(x3all));

xlabel('n');

ylabel('x3\_a\_l\_l');

1. According to the definition, x3\_all is the same as x3. Since x3 is real, then x3\_all is real.

****

**Matlab code:**

n0=0:31;

x3=[ones(1,8),zeros(1,24)];

a3=(1/32)\*fft(x3);

x32=zeros(1,32);

x38=zeros(1,32);

x312=zeros(1,32);

x3all=zeros(1,32);

for n=0:31

for k=-2:-1

x32(n+1)=x32(n+1)+conj(a3(-k+1)).\*exp(1i\*k\*n\*(2\*pi/32));

end

for k=0:2

x32(n+1)=x32(n+1)+a3(k+1).\*exp(1i\*k\*n\*(2\*pi/32));

end

end

for n=0:31

for k=-8:-1

x38(n+1)=x38(n+1)+conj(a3(-k+1)).\*exp(1i\*k\*n\*(2\*pi/32));

end

for k=0:8

x38(n+1)=x38(n+1)+a3(k+1).\*exp(1i\*k\*n\*(2\*pi/32));

end

end

for n=0:31

for k=-12:-1

x312(n+1)=x312(n+1)+conj(a3(-k+1)).\*exp(1i\*k\*n\*(2\*pi/32));

end

for k=0:12

x312(n+1)=x312(n+1)+a3(k+1).\*exp(1i\*k\*n\*(2\*pi/32));

end

end

for n=0:31

for k=-15:-1

x3all(n+1)=x3all(n+1)+conj(a3(-k+1))\*exp(1i\*k\*n\*(2\*pi/32));

end

for k=0:16

x3all(n+1)=x3all(n+1)+a3(k+1)\*exp(1i\*k\*n\*(2\*pi/32));

end

end

stem(n0,x3-x32);

xlabel('n');

ylabel('x3\_2');

hold on;

stem(n0,x3-x38);

xlabel('n');

ylabel('x3\_8');

hold on;

stem(n0,x3-x312);

xlabel('n');

ylabel('x3\_1\_2');

hold on;

stem(n0,x3-x3all,'\*');

xlabel('n');

ylabel('x3\_a\_l\_l');

As the figure shows, when more DTFS coefficients are contained, the figure converges to zero, which means the signals converge to x3[n]. that shows the Gibb’s phenomenon.

**Advanced problem**

Matlab code:

function a=dtfs(x,n\_init)

a=zeros(1,length(x);

if n\_init==0

for k=1:length(x)

for n=1:length(x)

a(k)=a(k)+x(n)\*exp(-1i\*(k-1)\*(n-1)\*(2\*pi/length(x));

end

end

elseif n\_init>0

x=[x(length(x)-n\_init+1:length(x)) x(1:length(x)-n\_init)];

for k=1:length(x)

for n=1:length(x)

a(k)=a(k)+x(n)\*exp(-1i\*(k-1)\*(n-1)\*(2\*pi/length(x));

end

end

else

x=[x(1-n\_init:length(x)) x(1:-n\_init)];

for k=1:length(x)

for n=1:length(x)

a(k)=a(k)+x(n)\*exp(-1i\*(k-1)\*(n-1)\*(2\*pi/length(x));

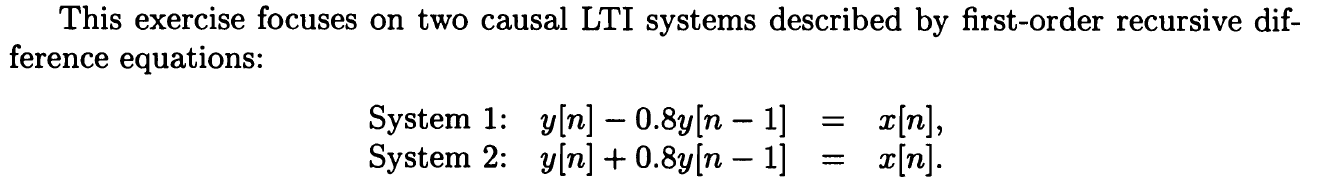
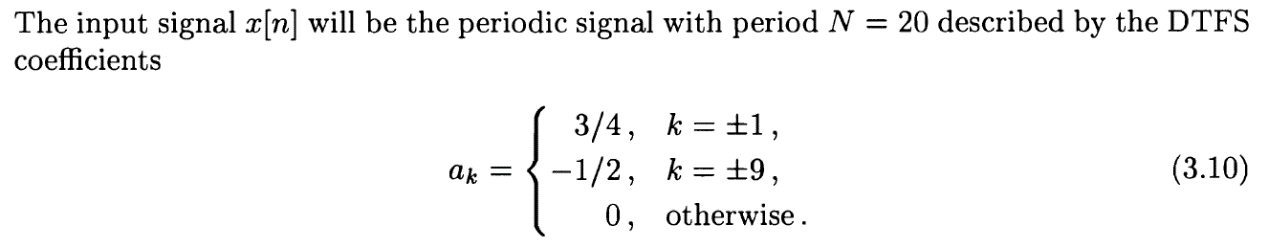
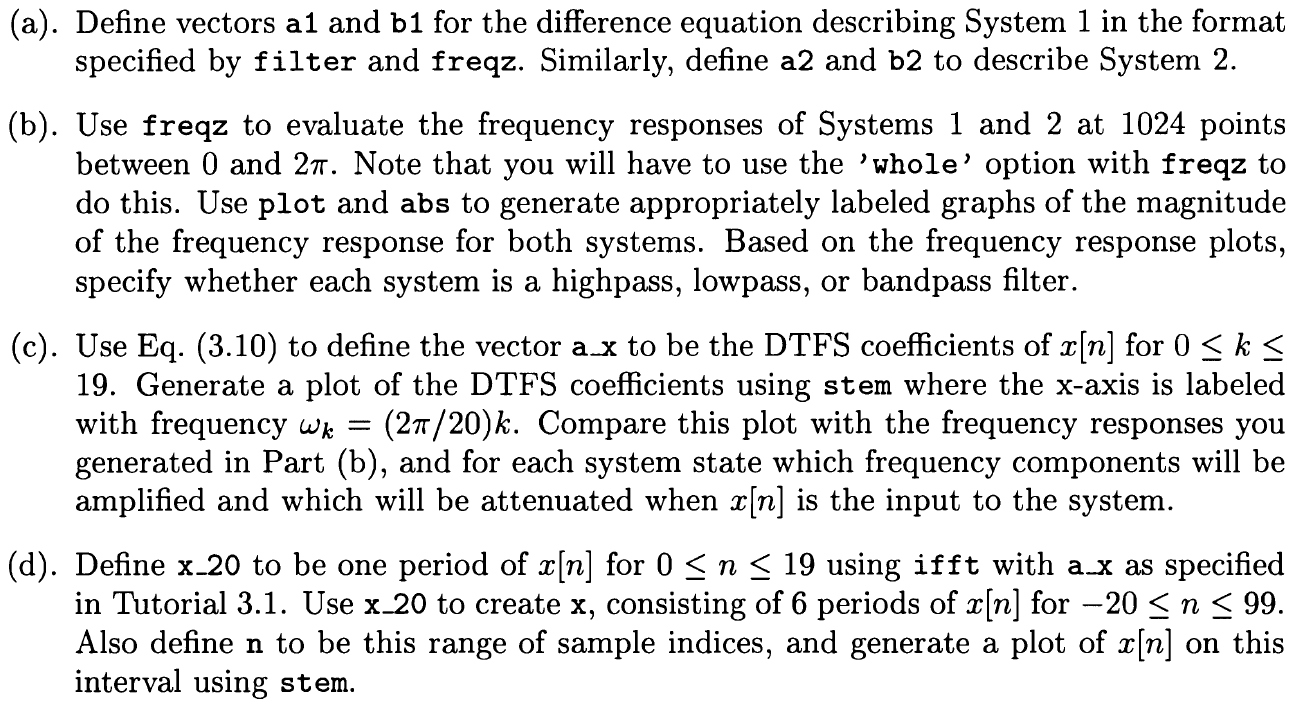
end

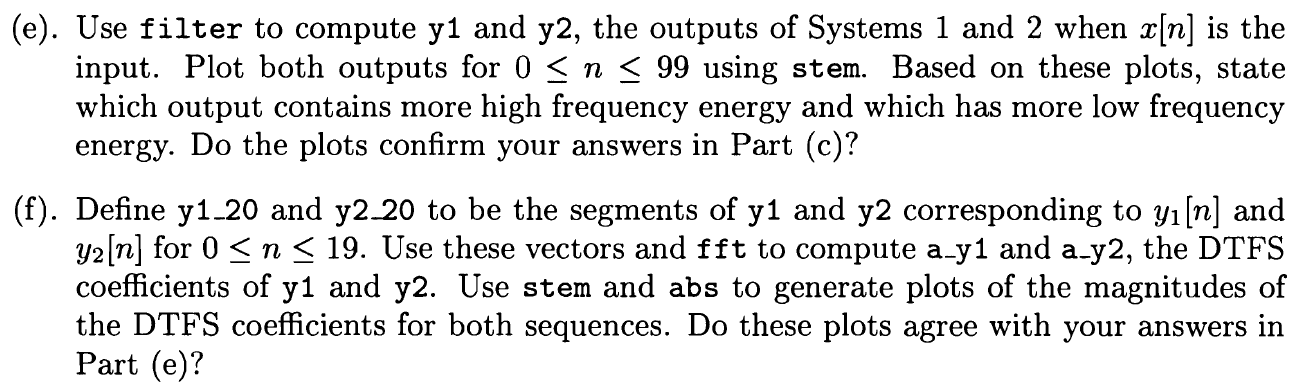
end

end

a=a/length(x);

**3.8**





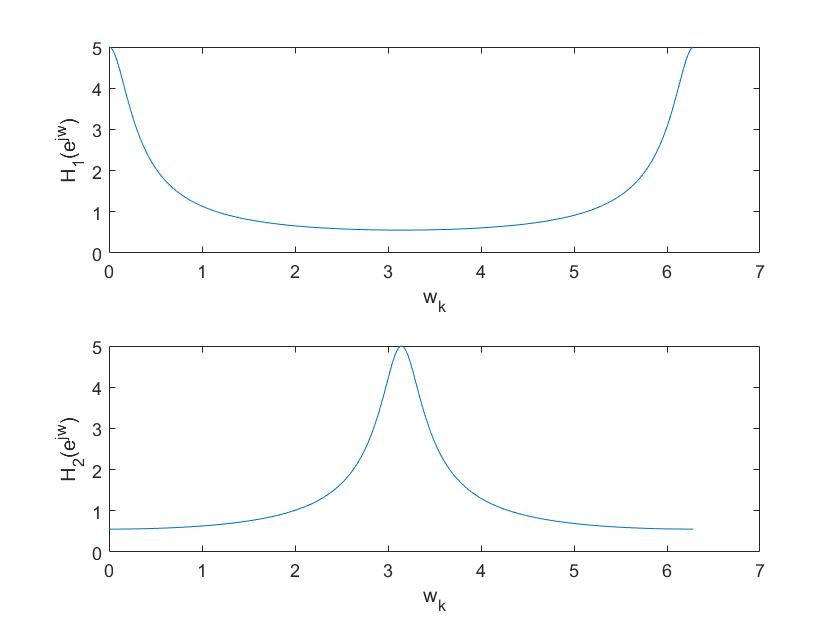
**Solution:**

1. a1=[1,-0.8]

b1=[1]

a2=[1,0.8]

b2=[1]



From the graph, it can be noticed that System 1 is lowpass filter while System 2 is highpass filter.

**Matlab code:**

n=0:2\*pi/1023:2\*pi;

a1=[1,-0.8];

b1=[1];

a2=[1,0.8];

b2=[1];

[H1,omega1]=freqz(b1,a1,1024,'whole');

[H2,omega2]=freqz(b2,a2,1024,'whole');

subplot(2,1,1);

plot(n,abs(H1));

xlabel('w\_k');

ylabel('H\_1(e^j^w)');

ylim([0 5]);

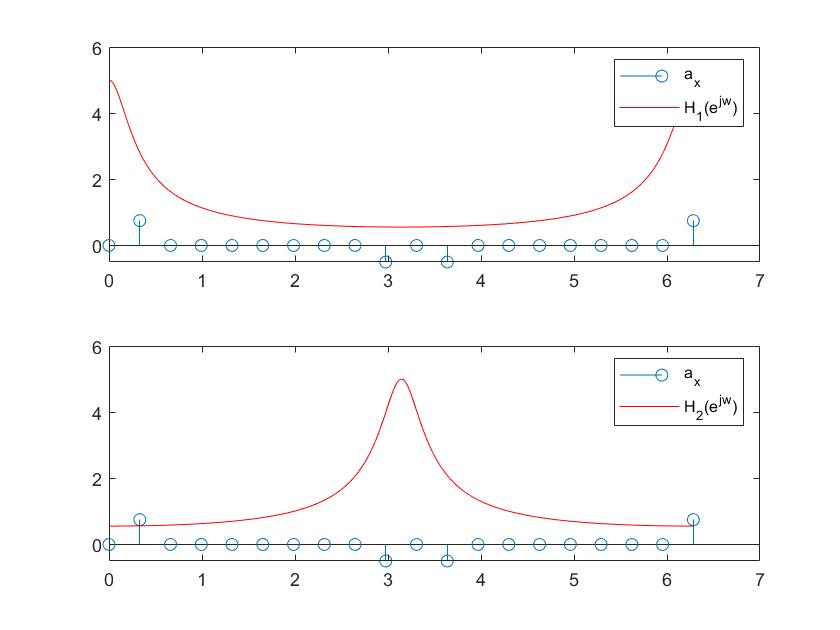
subplot(2,1,2);

plot(n,abs(H2));

xlabel('w\_k');

ylabel('H\_2(e^j^w)');

ylim([0 5]);



**Matlab code:**

n=0:2\*pi/1023:2\*pi;

a1=[1,-0.8];

b1=[1];

a2=[1,0.8];

b2=[1];

a\_x=[0,3/4,zeros(1,7),-1/2,0,-1/2,zeros(1,7),3/4];

[H1,omega1]=freqz(b1,a1,1024,'whole');

[H2,omega2]=freqz(b2,a2,1024,'whole');

subplot(2,1,1);

stem(0:2\*pi/19:2\*pi,a\_x);

hold on;

plot(n,abs(H1),'r');

legend('a\_x','H\_1(e^j^w)');

subplot(2,1,2);

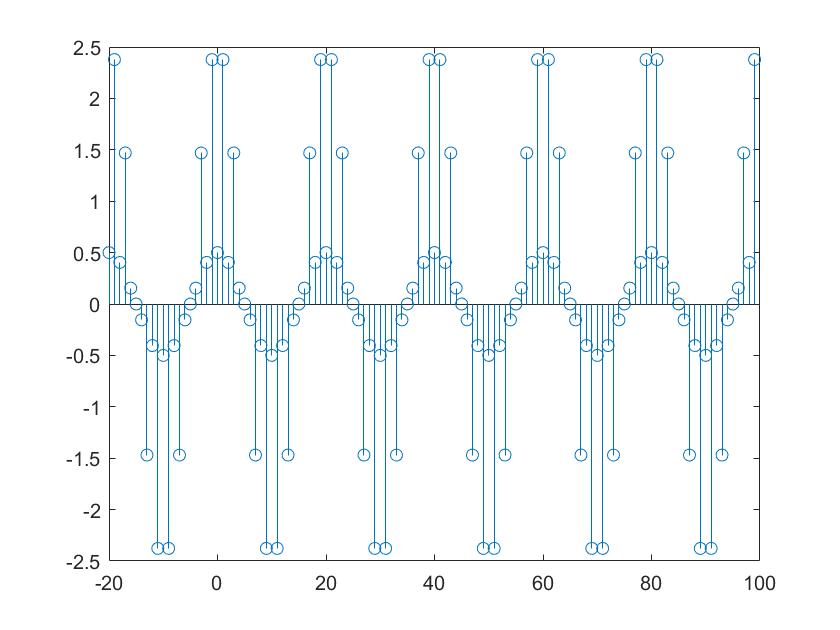
stem(0:2\*pi/19:2\*pi,a\_x);

hold on;

plot(n,abs(H2),'r');

legend('a\_x','H\_2(e^j^w)');

As the figure shows, in System 1, the 2nd and the 20th frequency components will be amplified and the 10th and the 12th frequency components will be attenuated. In system 2, the 10th and the 12th frequency components will be amplified, and the 2nd and the 20th frequency components will be attenuated.



The figure is shown above.

**Matlab code:**

n=-20:99;

a1=[1,-0.8];

b1=[1];

a2=[1,0.8];

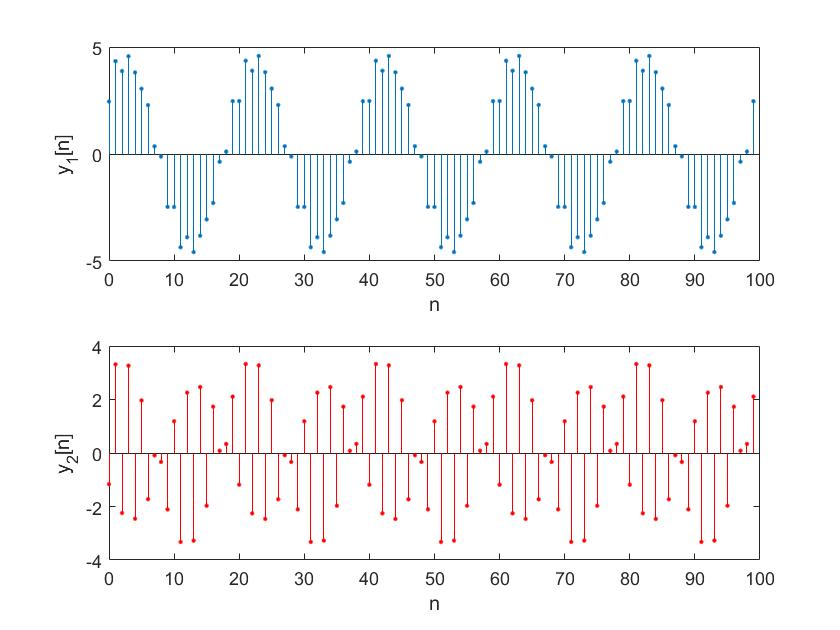
b2=[1];

a\_x=[0,3/4,zeros(1,7),-1/2,0,-1/2,zeros(1,7),3/4];

x\_20=20\*ifft(a\_x);

x=[x\_20,x\_20,x\_20,x\_20,x\_20,x\_20];

stem(n,x);

****

**Matlab code:**

n0=-20:99;

n=0:99;

a1=[1,-0.8];

b1=[1];

a2=[1,0.8];

b2=[1];

a\_x=[0,3/4,zeros(1,7),-1/2,0,-1/2,zeros(1,7),3/4];

x\_20=20\*ifft(a\_x);

x=[x\_20,x\_20,x\_20,x\_20,x\_20,x\_20];

y1=filter(b1,a1,x);

y2=filter(b2,a2,x);

y1\_1=y1(21:120);

y2\_1=y2(21:120);

subplot(2,1,1);

stem(n,y1\_1,'.');

xlabel('n');

ylabel('y\_1[n]');

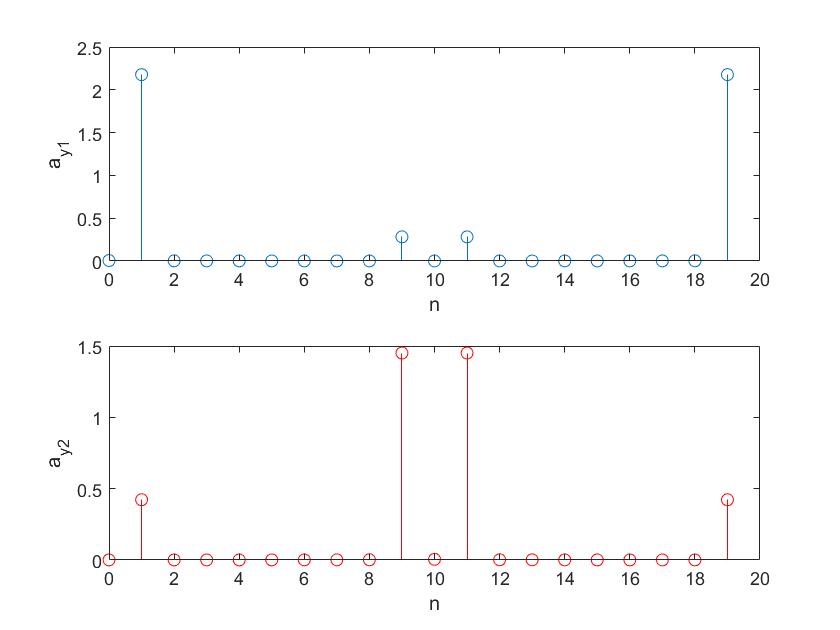
subplot(2,1,2);

stem(n,y2\_1,'r','.');

xlabel('n');

ylabel('y\_2[n]');

The figures of y1[n] and y2[n] are shown above. It can be noticed that y1[n] changes slowly and y2[n] changes quickly, that means y1[n] contains more low frequencies components and y2[n] contains more high frequencies components, which is confirmed in part (c).

****

**Matlab code:**

n0=-20:99;

n=0:19;

a1=[1,-0.8];

b1=[1];

a2=[1,0.8];

b2=[1];

a\_x=[0,3/4,zeros(1,7),-1/2,0,-1/2,zeros(1,7),3/4];

x\_20=20\*ifft(a\_x);

x=[x\_20,x\_20,x\_20,x\_20,x\_20,x\_20];

y1=filter(b1,a1,x);

y2=filter(b2,a2,x);

y1\_20=y1(21:40);

y2\_20=y2(21:40);

a\_y1=1/20\*fft(y1\_20);

a\_y2=1/20\*fft(y2\_20);

subplot(2,1,1);

stem(n,abs(a\_y1));

xlabel('n');

ylabel('a\_y\_1');

subplot(2,1,2);

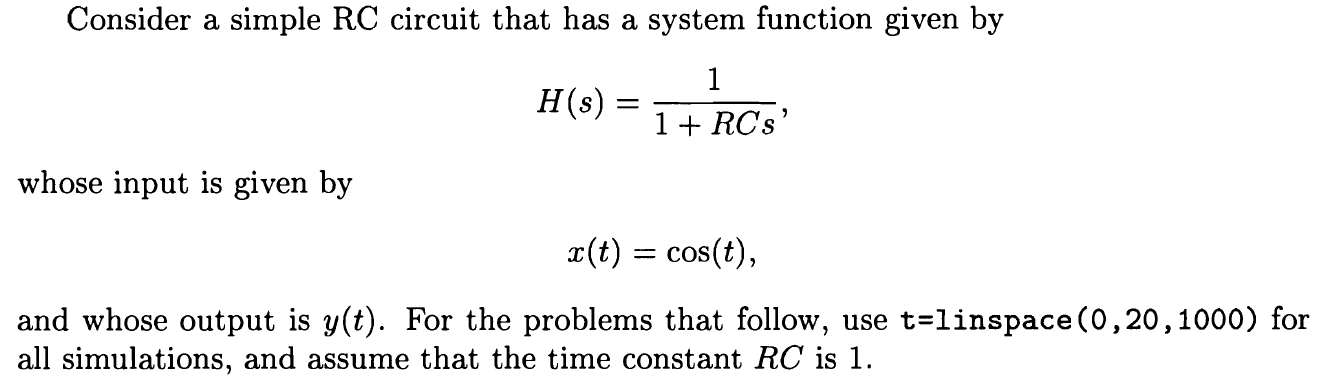
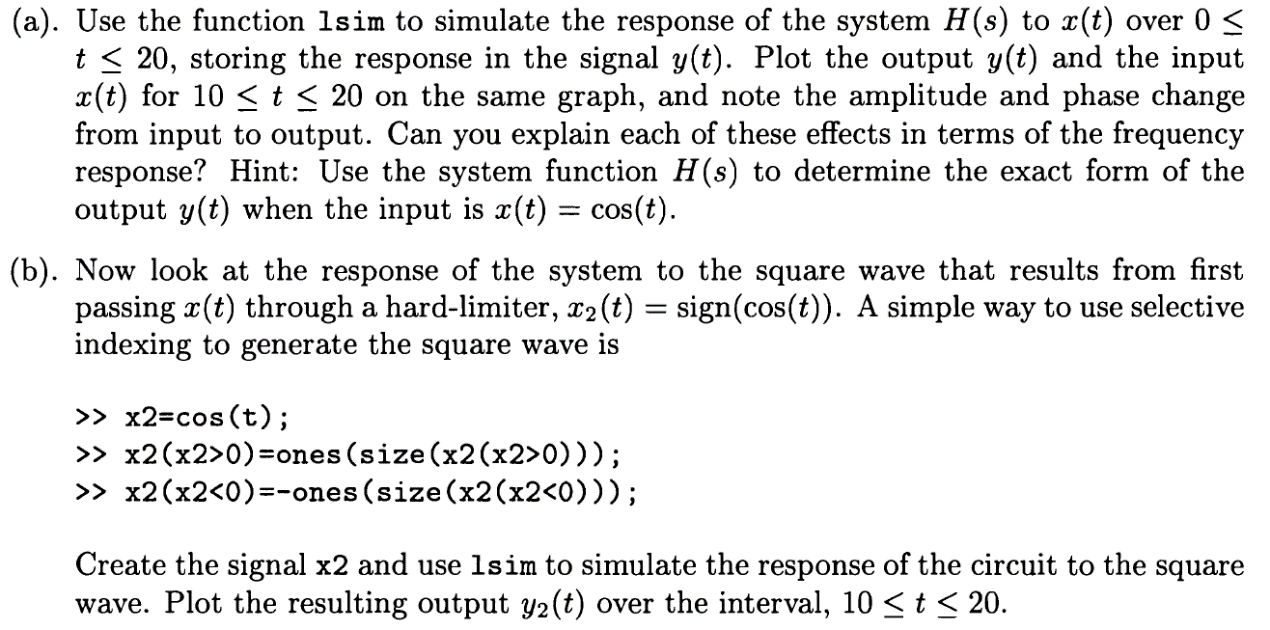
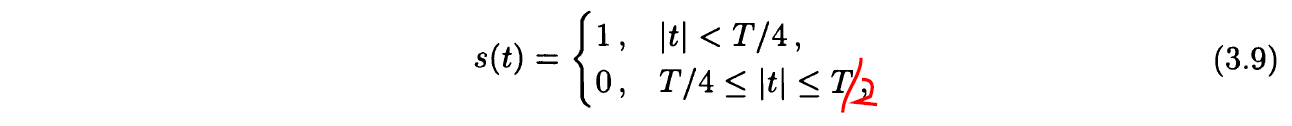
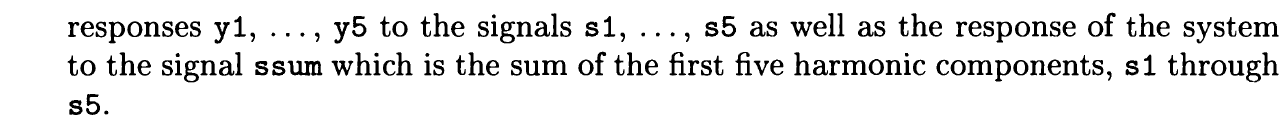
stem(n,abs(a\_y2),'r');

xlabel('n');

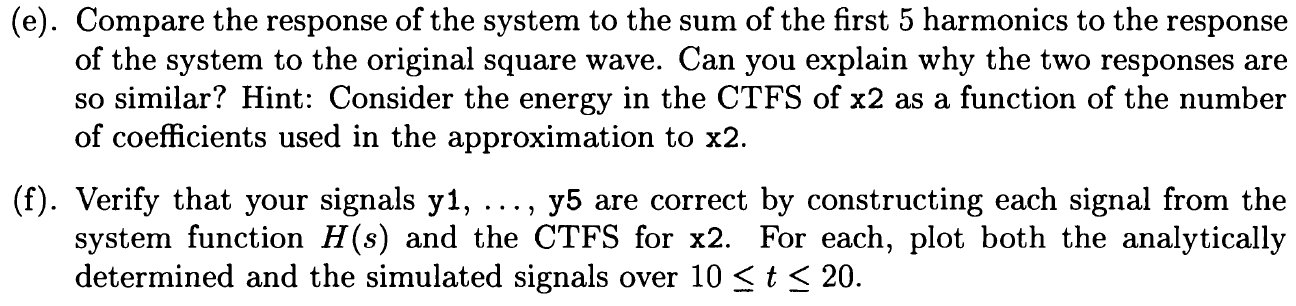
ylabel('a\_y\_2');

The plots are shown above, and it agrees with the result in part(e).

**3.9**

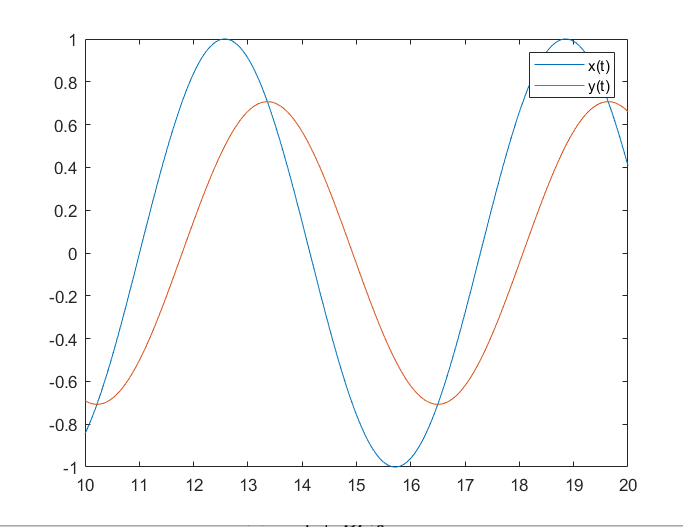






**Solution:**

**（a）**



|H (s)|=|1/(1+s)|=<1. so the peak will drop.But the phase will not change.

**Matlab code:**

%3.9(a)

t=linspace(0,20,1000);

x=cos(t);

b=1;

a=[1,1];

y=lsim(b,a,x,t);

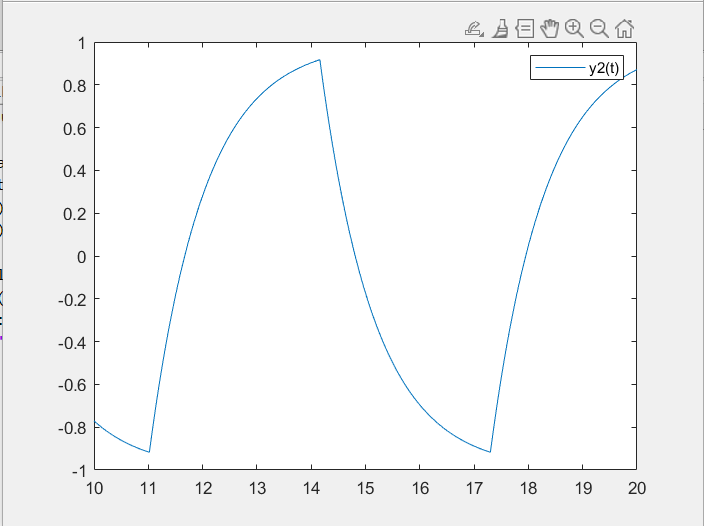
plot (10:0.02:20,x(1,500:1000));

hold on;

plot(10:0.02:20,y(500:1000,1)');

legend('x(t)','y(t)')

（b）



**Matlab code:**

%3.9(b)

t=linspace(0,20,1000);

x2=cos(t);

x2(x2>0)=ones(size(x2(x2>0)));

x2(x2<0)=-ones(size(x2(x2<0)));

b=1;

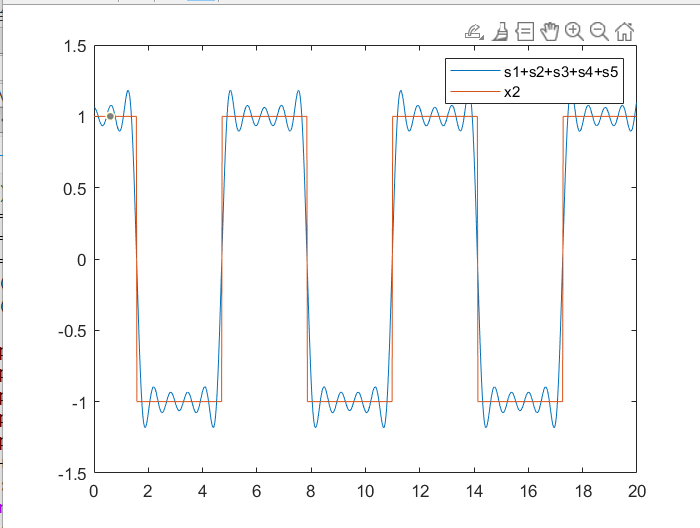
a=[1,1];

y2=lsim(b,a,x2,t);

plot(10:0.02:20,y2(500:1000));

legend('y2(t)');

（c）



From equation 3.9, we can get x2(t)=s(t)-s(t-π).

From s1=apos-k(1)\*exp(j\*t)+aneg-k(1)\*exp(-j\*t)

We can calculate s2 s3 s4 s5

**Matlab code:**

%3.9(c)

A1\_k=1:10;

A2\_k=1:10;

for i = 1:10

A1\_k(i) = 2 \* sin(pi\*i/2)/(pi\*i);

A2\_k(i) = 2 \* sin(pi\*(-i)/2)/(pi\*(-i));

end

s1 = A1\_k(1)\*exp(1\*1i\*t)+A2\_k(1)\*exp((-1)\*1i\*t);

s2 = A1\_k(3)\*exp(3\*1i\*t)+A2\_k(3)\*exp((-3)\*1i\*t);

s3 = A1\_k(5)\*exp(5\*1i\*t)+A2\_k(5)\*exp((-5)\*1i\*t);

s4 = A1\_k(7)\*exp(7\*1i\*t)+A2\_k(7)\*exp((-7)\*1i\*t);

s5 = A1\_k(9)\*exp(9\*1i\*t)+A2\_k(9)\*exp((-9)\*1i\*t);

ss =s1+s2+s3+s4+s5;

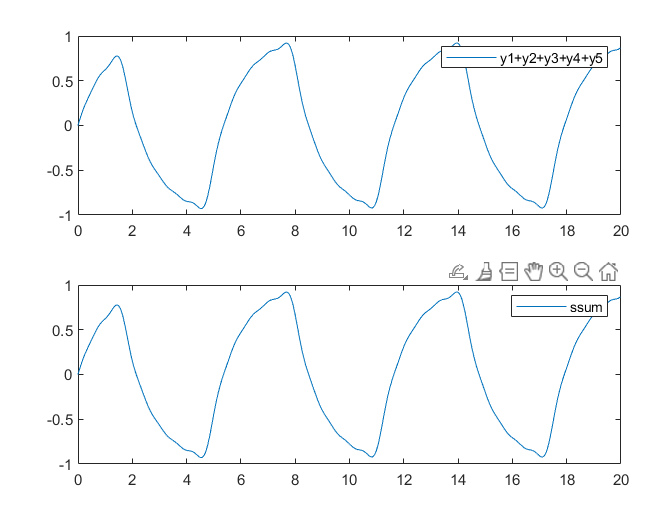
plot(0:0.02:19.98,ss)

hold on

plot(0:0.02:19.98,x2)

legend('s1+s2+s3+s4+s5','x2')

（d）



Obviously, the two pictures are the same, because the circuit response is linear.

**Matlab code:**

%3.9(d)

y1=lsim(b,a,s1,t);

y2=lsim(b,a,s2,t);

y3=lsim(b,a,s3,t);

y4=lsim(b,a,s4,t);

y5=lsim(b,a,s5,t);

ssum=lsim(b,a,ss,t);

subplot(2,1,1);

plot(0:0.02:19.98,y1+y2+y3+y4+y5);

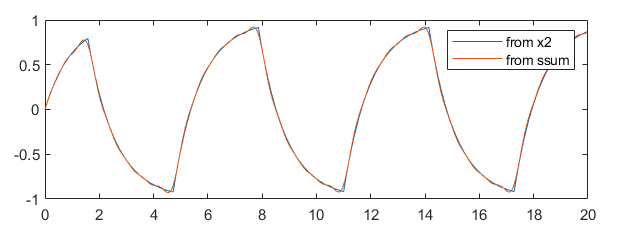
legend('y1+y2+y3+y4+y5')

subplot(2,1,2);

plot(0:0.02:19.98,ssum);

legend('ssum')

(e)



With the increase of k, the influence of sk on the whole decomposition is smaller, and x2 can be reflected to a great extent when k=5, so the responses of the two are very close.

**Matlab code:**

yx=lsim(b,a,x2,t);

figure(1);

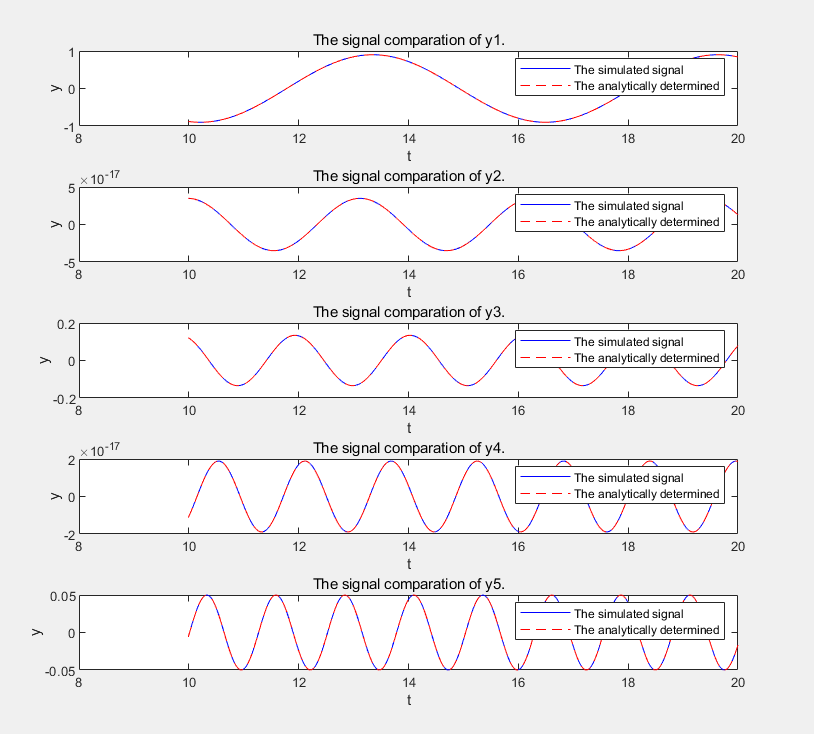
plot(0:0.02:19.98,yx);

hold on

plot(0:0.02:19.98,ssum);

legend('from x2','from ssum')

(f)



**Matlab code:**

%3.9(f)

for n=1:5

k=2\*n-1;

a\_x2=2\*sin(pi\*k/2)/pi./k;

end

A1=(1/(1+1i))\*a\_x2(1)\*exp(1j\*t)+(1/(1-1i))\*a\_x2(1)\*exp(-1j\*t);

A2=(1/(1+3j))\*a\_x2(2)\*exp(3j\*t)+(1/(1-3j))\*a\_x2(2)\*exp(-3j\*t);

A3=(1/(1+5j))\*a\_x2(3)\*exp(5j\*t)+(1/(1-5j))\*a\_x2(3)\*exp(-5j\*t);

A4=(1/(1+7j))\*a\_x2(4)\*exp(7j\*t)+(1/(1-7j))\*a\_x2(4)\*exp(-7j\*t);

A5=(1/(1+9j))\*a\_x2(5)\*exp(9j\*t)+(1/(1-9j))\*a\_x2(5)\*exp(-9j\*t);

S1=lsim(sys,s1,t);

S2=lsim(sys,s2,t);

S3=lsim(sys,s3,t);

S4=lsim(sys,s4,t);

S5=lsim(sys,s5,t);

subplot(5,1,1);

plot(t,A1),xlim([10,20]);

hold on ;

plot(t,S1),xlim([10,20]);

legend('The analytically determined','The simulated signal')

subplot(5,1,2);

plot(t,A2),xlim([10,20]);

hold on ;

plot(t,S2),xlim([10,20]);

legend('The analytically determined','The simulated signal')

subplot(5,1,3);

plot(t,A3),xlim([10,20]);

hold on ;

plot(t,S3),xlim([10,20]);

legend('The analytically determined','The simulated signal')

subplot(5,1,4);

plot(t,A4),xlim([10,20]);

hold on ;

plot(t,S4),xlim([10,20]);

legend('The analytically determined','The simulated signal')

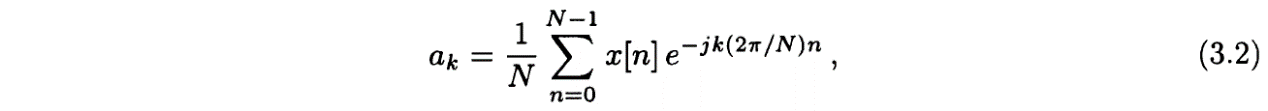
subplot(5,1,5);

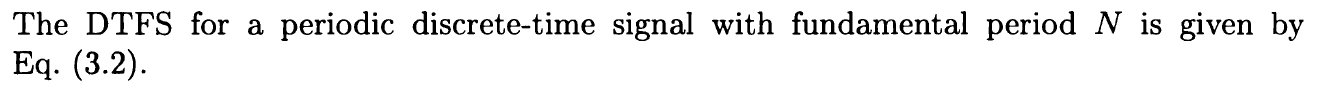
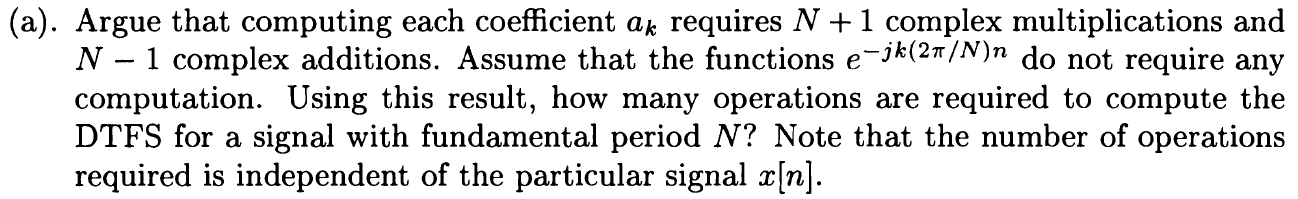
plot(t,A5),xlim([10,20]);

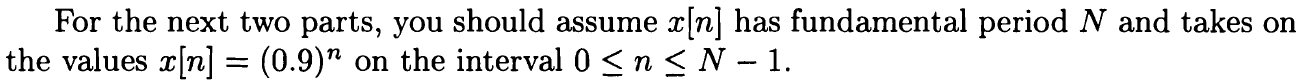
hold on ;

plot(t,S5),xlim([10,20]);

legend('The analytically determined','The simulated signal')

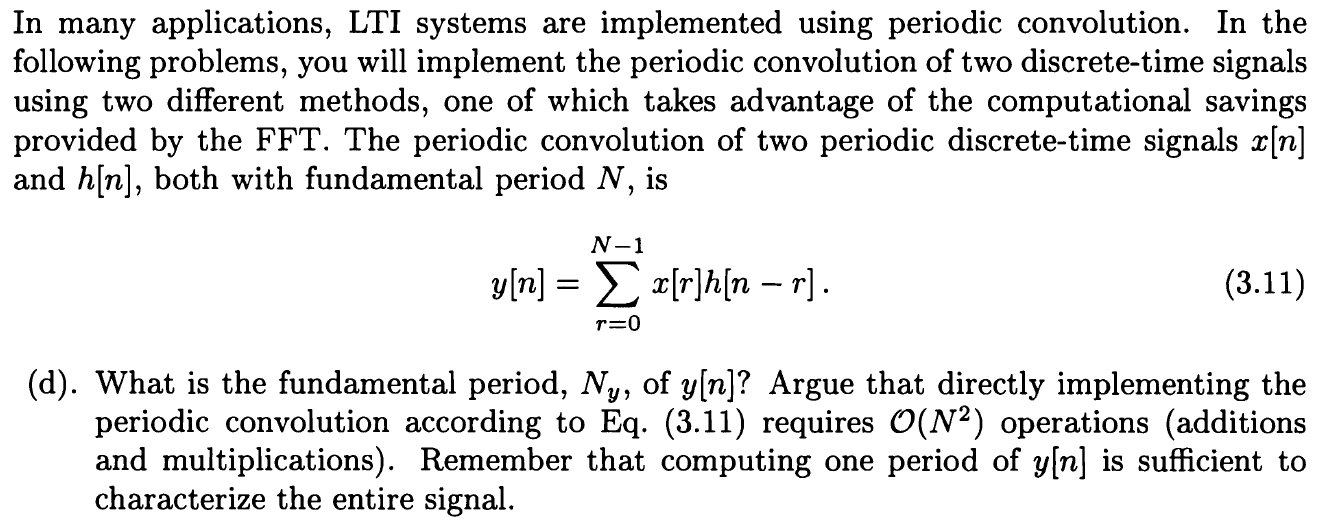
**3.10**





(b). If you have not already done the Advanced Problem in Exercise 3.5 writing **dtfs**, do so now. You will compare the computation time this algorithm requires with that required by **fft**. The MATLAB command **tic** and **toc** can be used to measure the time for computation. Find the time for computation using DTFS for , save these values in the vector **dtfstime**.

(c). Now, compute the DTFS coefficients of x[n] for using **fft**. Find the time of computation for each N and store the values in the vector **ffttime**. Plot **dtfstime** and **ffttime** versus N using **loglog**. How does the time required by **fft** compare to that required by **DTFS**, particularly for large values of N?

The computational savings is defined as the ratio of the time required by the slow algorithm to those of the fast algorithm.

(e). Assume both x[n] and h[n] have fundamental period N = 32, and are given by and over the interval . Compute the periodic convolution of x[n] with h[n] and plot y[n] for . Store the time of computation, required to implement the periodic convolution in **convtime**. Remember to call **tic** after creating x and h, the vectors representing x[n] and h[n]. Hint: To implement the periodic convolution, first store x[n] and h[n] over the interval in the row vectors x and h, respectively. Call **tic** and then use **conv([x x], h)**. The periodic convolution can be extracted from a portion of this signal.

(f). Repeat Part (e) for N = 64, 128, again plotting a period of y[n] and storing the computation time in **convtime.**



(h). Repeat Part (g) for N = 64, 128, again plotting a period of y[n] and storing the computation time in **iffttime.** Again check the validity of your implementation by comparing y[n] with that computed in Part (f).

(i). Compute the **convtime** and **iffttime** for Compute the ratios of **convtime** to **iffttime** for , then plot it versus N using **semilogx.** Which method of convolution is more efficient for large values of N? Justify your answer.

**Solution:**

**(a)**

The DTFS coefficients has the same period as x[n], then we only need to compute within one period N. Therefore, only N samples in are needed. Since computing each coefficient requires N+1 complex multiplications and N-1 complex additions, N(N+1) complex multiplications and N(N-1) complex additions are needed for N .

**(b)**

According to the calculation, when n is equal to [8 32 64 128 256], the running time is respectively 0.000166800000000000 0.000204400000000000 0.000657400000000000 0.00306140000000000 0.0119028000000000

**Matlab code:**

i=1;

for N=[8 32 64 128 256]

x=(0.9).^[0:N-1];

tic;

X=dtfs(x,0);

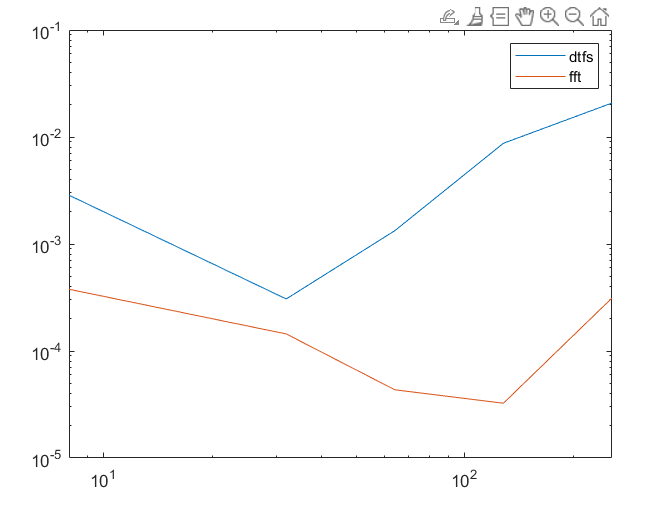
c=toc;

dtfscomps(i)=c;

i=i+1;

end

**(c)**



**Matlab code:**

%3.10(c)

M=[8 32 64 128 256];

i=1;

for N=[8 32 64 128 256]

x=(0.9).^[0:N-1];

tic;

X=dtfs(x,0);

c=toc;

dtfscomps(i)=c;

i=i+1;

end

j=1;

for G=[8 32 64 128 256]

x=(0.9).^[0:G-1];

tic;

Y=fft(x)/G;

c=toc;

fftcomps(j)=c;

j=j+1;

end

figure(1)

loglog(M, dtfscomps);

hold on;

loglog(M, fftcomps);

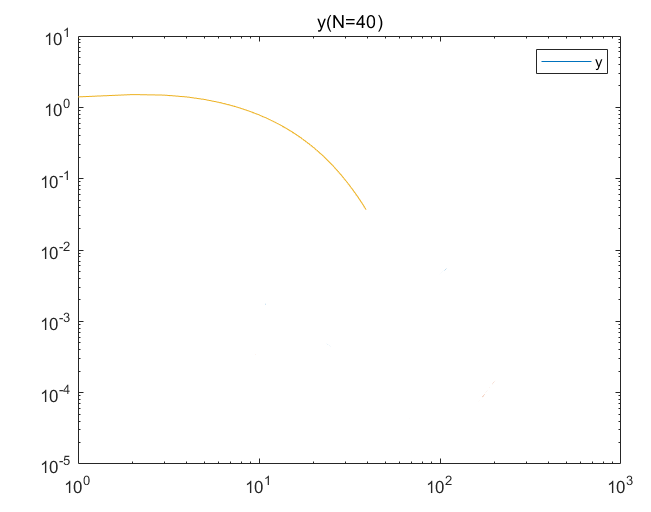
legend('dtfs','fft');

**(d)**

The fundamental period .

we can easily get that .

**(e)**



**Matlab code:**

n=0:39;

x=0.9.^n;

h=0.5.^n;

tic;

y11=conv([x,x],h)

f40c=toc;

y22=y11(1:40);

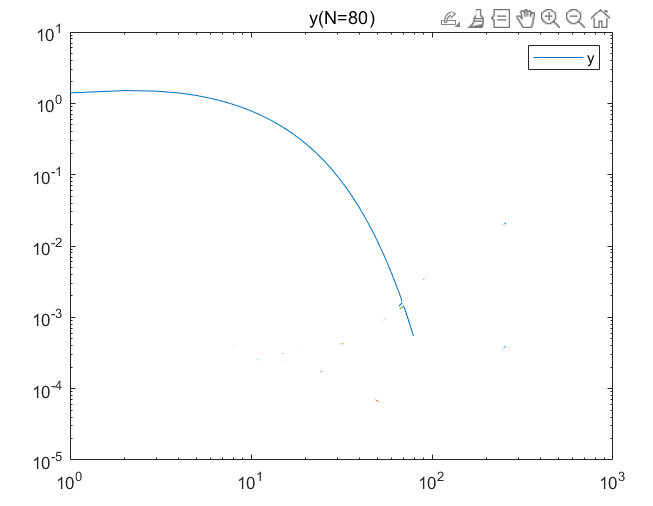
% figure(2);

plot(n,y22);

legend('y')

title('y(N=40)')

**(f)**

****

**Matlab code:**

%3.10(f)

n=0:79;

x=0.9.^n;

h=0.5.^n;

tic;

y1=conv([x,x],h)

f80c=toc;

y2=y1(1:80);

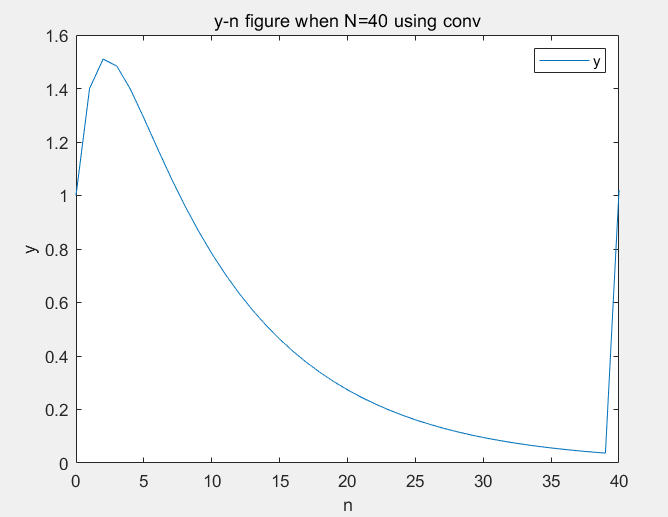
% figure(2);

plot(n,y2);

legend('y')

title('y(N=80)')

**(g)**



**Matlab code:**

n=0:39;

x=0.9.^n;

h=0.5.^n;

ax=fft(x);

ah=fft(h);

tic;

ay=ax.\*ah;

y2=ifft(ay);

f40f=toc;

figure(5);

plot(n,y1(1:40));

hold on;

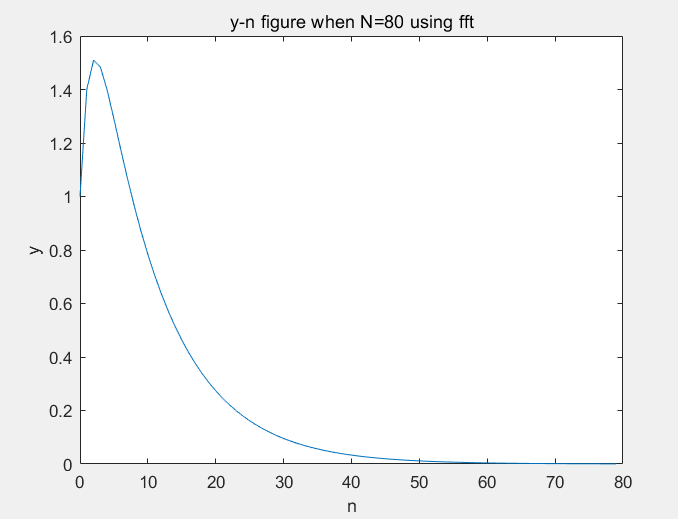
plot(n,y2(1:40),'--');

legend('y1','y2')

xlabel('n');

ylabel('y[n] (N=40)');

**(h)**



**Matlab code:**

%3.10(h)

n=0:79;

x=0.9.^n;

h=0.5.^n;

tic;

ax=fft(x);

ah=fft(h);

ay=ax.\*ah;

y2\_=ifft(ay);

f80f=toc;

figure(6);

plot(n,y1\_(1:80));

hold on;

plot(n,y2\_(1:80),'--');

legend('y1','y2')

xlabel('n');

ylabel('y[n] (N=80)');

**(i)**

But with the increase of n, the iff effect is better, so it should be selected when it is greater than 80.