Computer Assignment 2

TTT4115 Kommunikasjonsteori

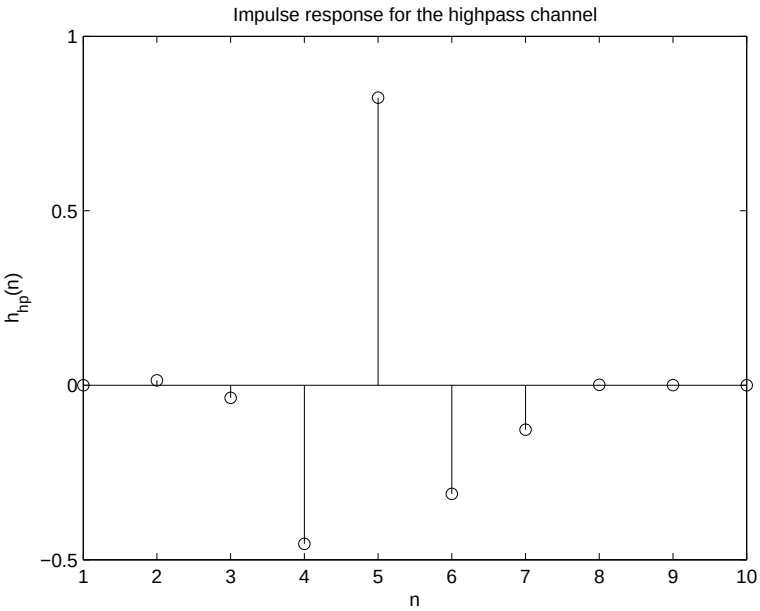
By Erik Moen, Geir Huth and Qingrui Zhou.

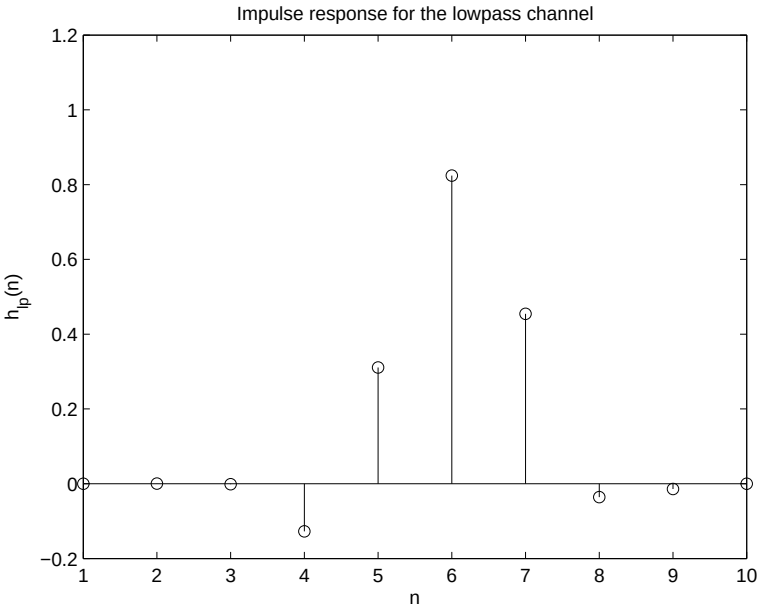
Contents

Oppgave 1 a)

Impulse responses for the high- and lowpass channels.

```
clear;
close all;
format compact;
format long g;
defaultStream = RandStream.getDefaultStream();
reset(defaultStream);

k = tan(pi.*[0.38,0.4,0.46,0.49]);
imphp = synN(1,0,k);
implp = synN(0,1,k);

figure(); stem(imphp); xlabel('n'); ylabel('h_{hp}(n)');
  title('Impulse response for the highpass channel');
figure(); stem(implp); xlabel('n'); ylabel('h_{lp}(n)');
  title('Impulse response for the lowpass channel');
```



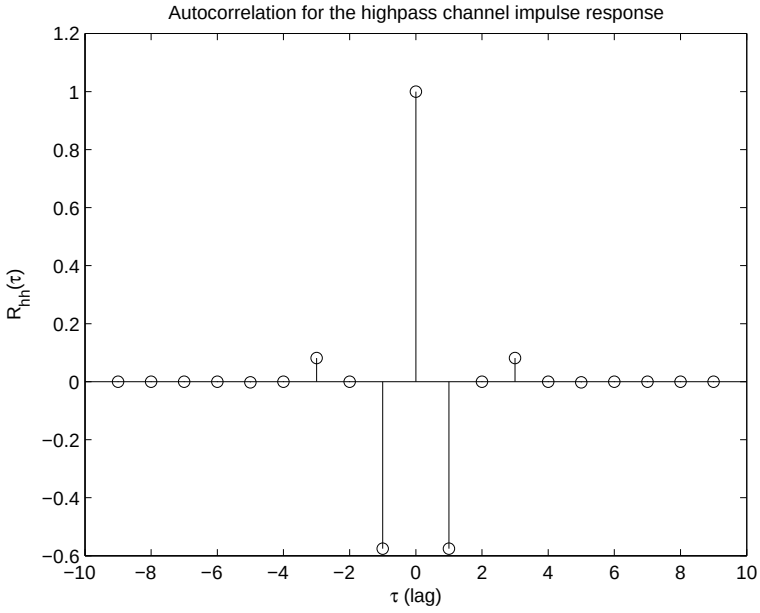


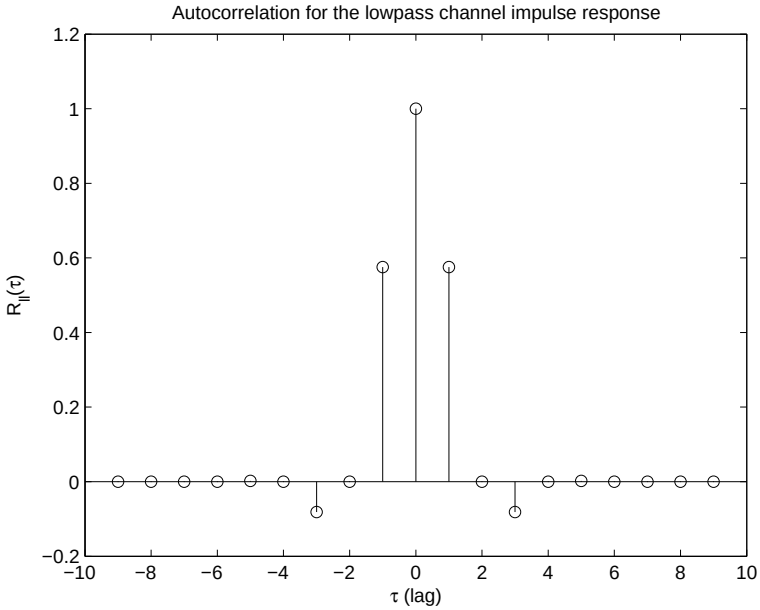
Oppgave 1 b)

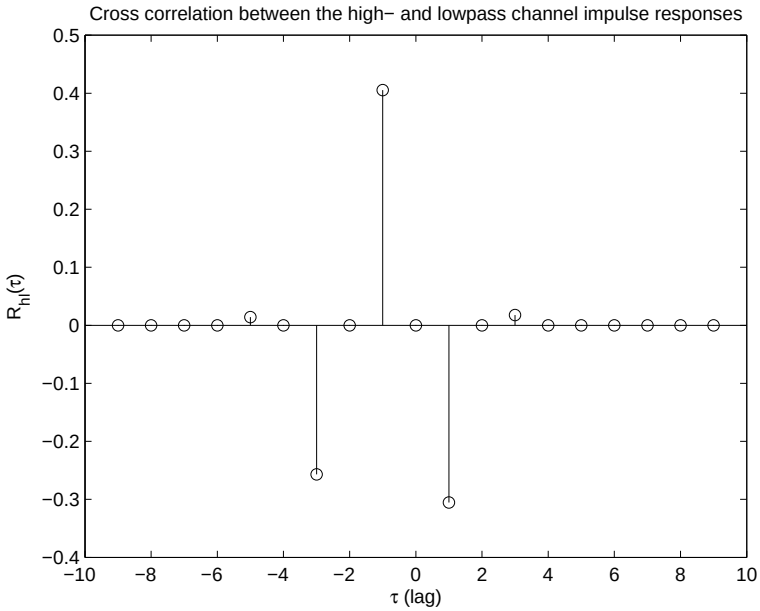
We show the impulses to be orthogonal at the required offsets, first visually.

```
[Rhh,thp]=xcorr(imphp,imphp);
```

```
[Rll,tlp]=xcorr(implp,implp);
[Rhl,thl]=xcorr(imphp,implp);

figure(); stem(thp,Rhh); xlabel('\tau (lag)'); ylabel('R_{hh}(\tau)');
title('Autocorrelation for the highpass channel impulse response');
figure(); stem(tlp,Rll); xlabel('\tau (lag)'); ylabel('R_{ll}(\tau)');
title('Autocorrelation for the lowpass channel impulse response');
figure(); stem(thl,Rhl); xlabel('\tau (lag)'); ylabel('R_{hl}(\tau)');
title('Cross correlation between the high- and lowpass channel impulse responses');
snapnow;
```







The maximal unwanted correlations are shown below. As we can see, there are only residual roundoff errors.

```
center = find(thp == 0);
idx=sort([center+(2:2:(center-1)), center-(2:2:(center-1))]);
max(abs(Rhh(idx)))
```

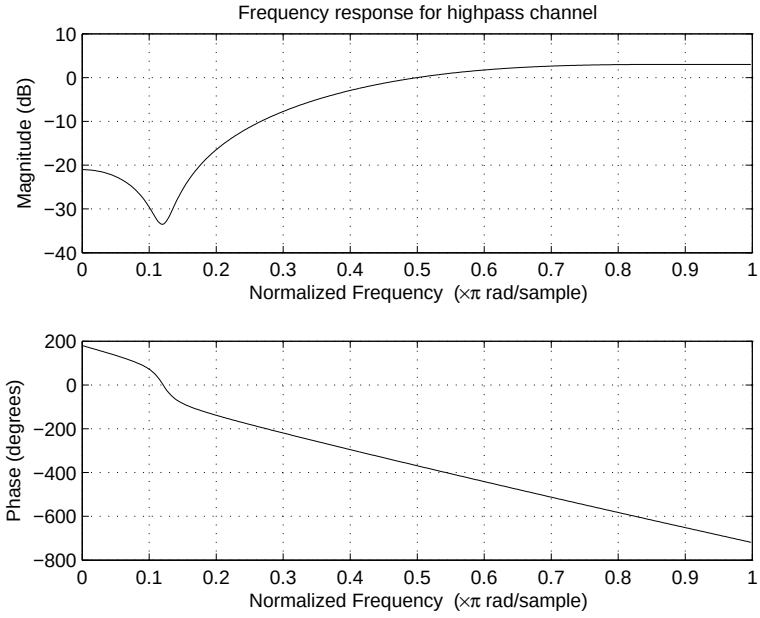
```
center = find(tlp == 0);
idx=sort([center+(2:2:(center-1)), center-(2:2:(center-1))]);
max(abs(Rll(idx)))
center = find(thl == 0);
idx=sort([center+(2:2:(center-1)), center-(2:2:(center-1)), center]);
max(abs(Rhl(idx)))
```

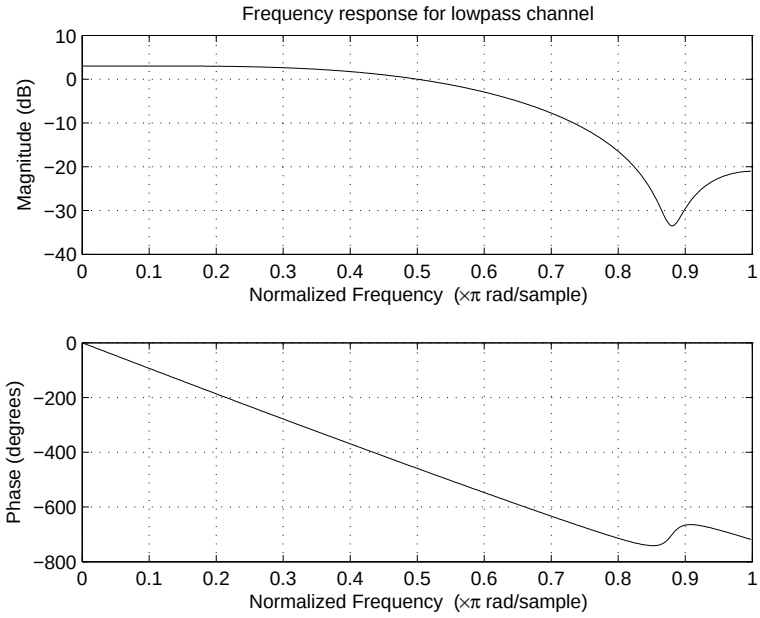
```
ans =
      1.11022302462516e-16
ans =
      5.55111512312578e-17
ans =
      3.46944695195361e-17
```

Oppgave 1 c)

The frequency responses for the two channels are shown.

```
figure(); freqz(imphp);
title('Frequency response for highpass channel');
figure(); freqz(implp);
title('Frequency response for lowpass channel');
```





Oppgave 1 d)

The variance over the channel is computed using two gaussian sources.

```
N = 10000;

% Genererer gaussisk sty
En1g = randn(1,N);
En2g = randn(1,N);

% Modulerer
Yn1 = synN(En1g,En2g,k);
var(Yn1)
```

```
ans =
         0.980976691960769
```

We observe the variance to be the same as the source signals, which makes sense as the rate is doubled.

Oppgave 1 e)

The variance over the channel is computed using two binary sources.

```
% Genererer binr gaussisk sty
En1b = (randi(2,1,N)-1.5)*2;
En2b = (randi(2,1,N)-1.5)*2;

% Modulerer
Yn2 = synN(En1b,En2b,k);
var(Yn2)
```

```
ans =
         0.999643668088163
```

We observe the variance to be the same as the source signals, which makes sense as the rate is doubled.

Oppgave 1 f)

We will show that the input signals can be reconstructed perfectly by subtracting the reconstructed output from the input.

Note that the modulation and demodulation creates an offset of 4 samples.

```
% Rekonstruerer signalene
[Xn1g,Xn2g] = anaN(Yn1,k);
[Xn1b,Xn2b] = anaN(Yn2,k);

% Finner maksimum differanse mellom innsignalene og de rekonstruerte
maxdiff_hpg = max( abs(En1g - Xn1g(5:N+4)) )
maxdiff_lpg = max( abs(En2g - Xn2g(5:N+4)) )
maxdiff_hpb = max( abs(En1b - Xn1b(5:N+4)) )
maxdiff_lpb = max( abs(En2b - Xn2b(5:N+4)) )
```

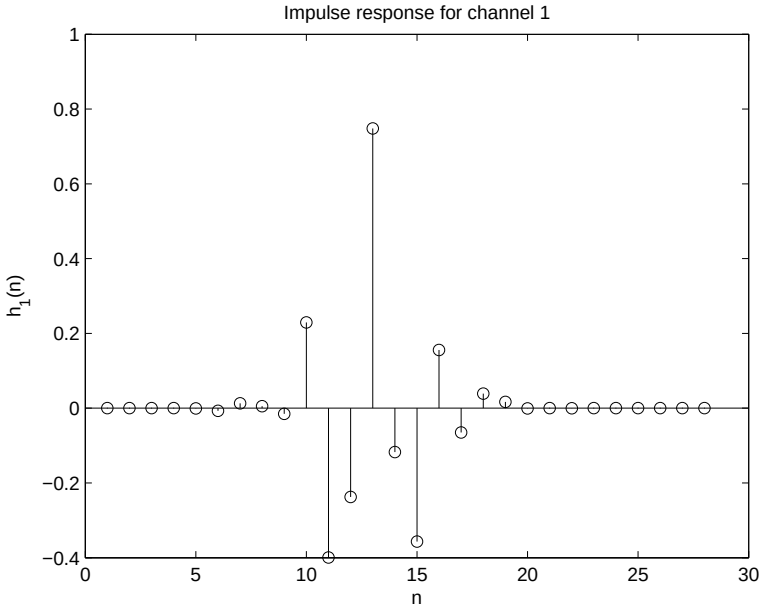
```
maxdiff_hpg =
      2.22044604925031e-15
maxdiff_lpg =
      1.77635683940025e-15
maxdiff_hpb =
      4.44089209850063e-16
maxdiff_lpb =
      4.44089209850063e-16
```

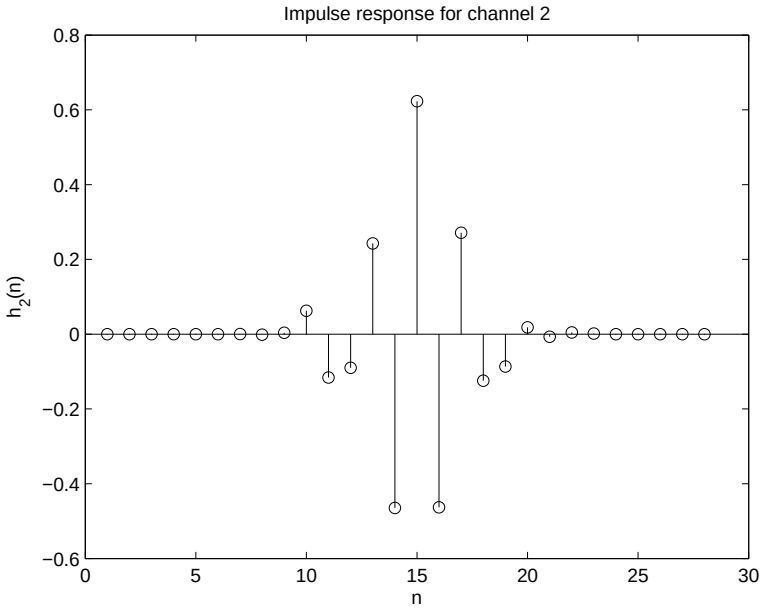
We attribute the differences to rounding errors.

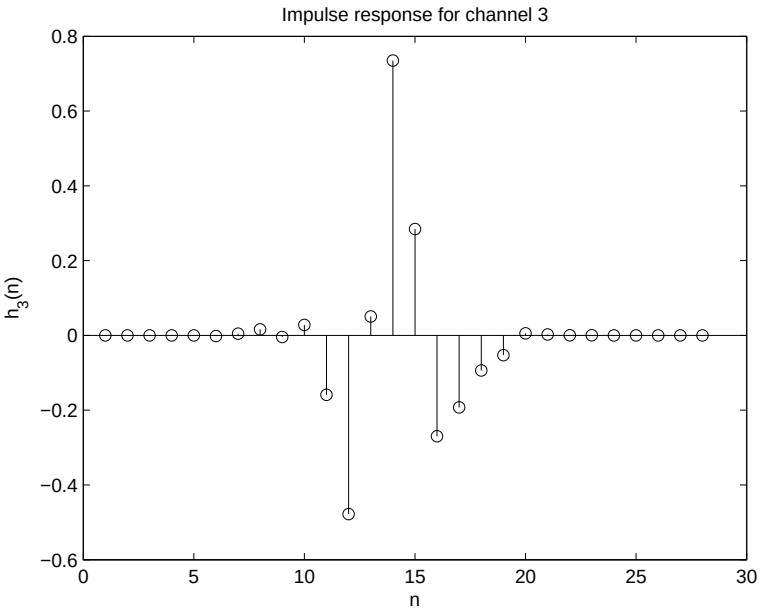
Oppgave 2 a)

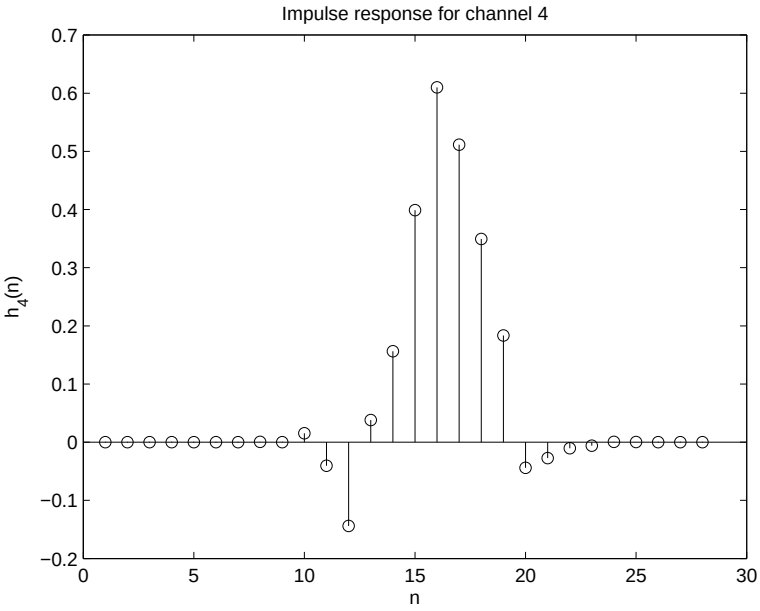
The impulse and frequency responses for the channels in a 4-channel transmitter are drawn.

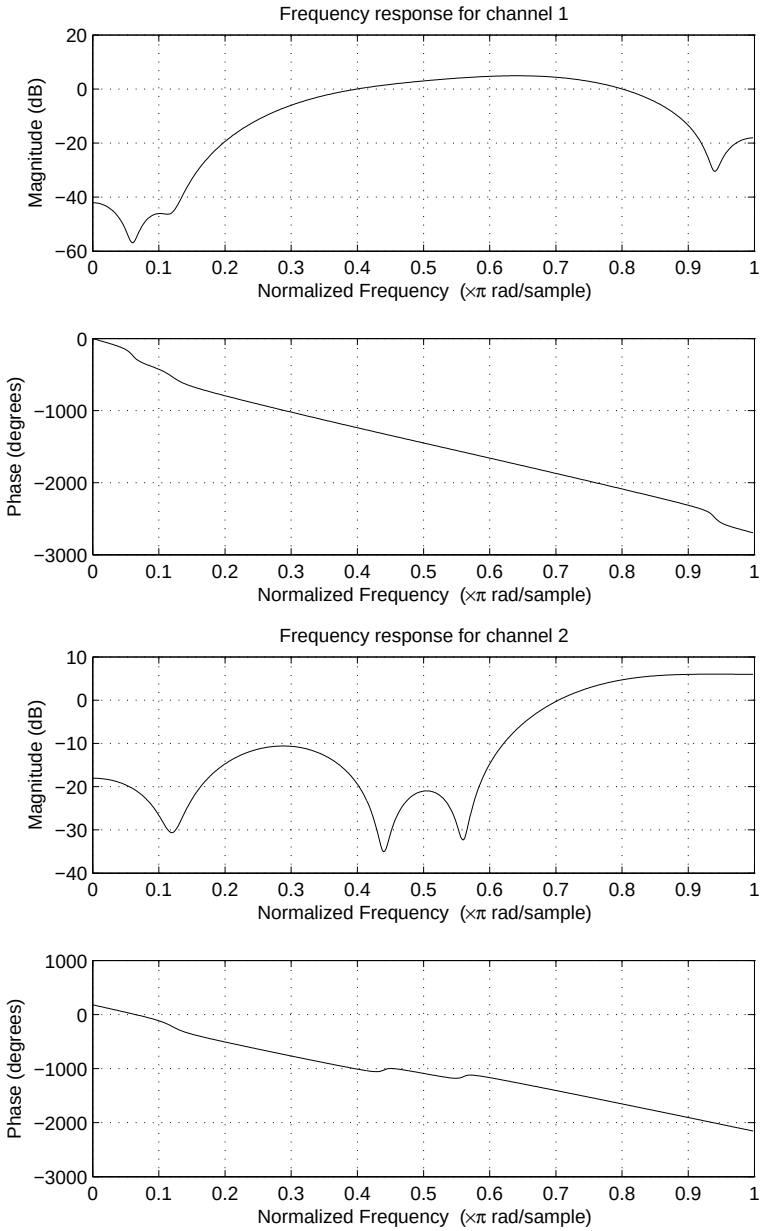
```
close all;

imp1 = synN( synN(1,0,k), synN(0,0,k) ,k)';
imp2 = synN( synN(0,1,k), synN(0,0,k) ,k)';
imp3 = synN( synN(0,0,k), synN(1,0,k) ,k)';
imp4 = synN( synN(0,0,k), synN(0,1,k) ,k)';
imp = [imp1, imp2, imp3, imp4];

figure(); stem(imp1); xlabel('n'); ylabel('h_1(n)');
title('Impulse response for channel 1');
figure(); stem(imp2); xlabel('n'); ylabel('h_2(n)');
title('Impulse response for channel 2');
figure(); stem(imp3); xlabel('n'); ylabel('h_3(n)');
title('Impulse response for channel 3');
figure(); stem(imp4); xlabel('n'); ylabel('h_4(n)');
title('Impulse response for channel 4');

figure(); freqz(imp1);
title('Frequency response for channel 1');
figure(); freqz(imp2);
title('Frequency response for channel 2');
figure(); freqz(imp3);
title('Frequency response for channel 3');
figure(); freqz(imp4);
title('Frequency response for channel 4');
```

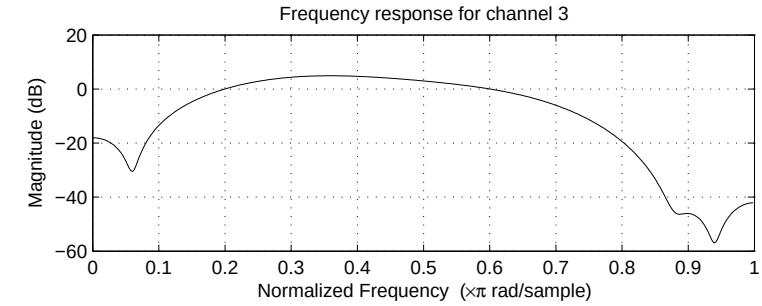

Impulse response for channel 1
h1(n)
n
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0
5
10
15
20
25
30

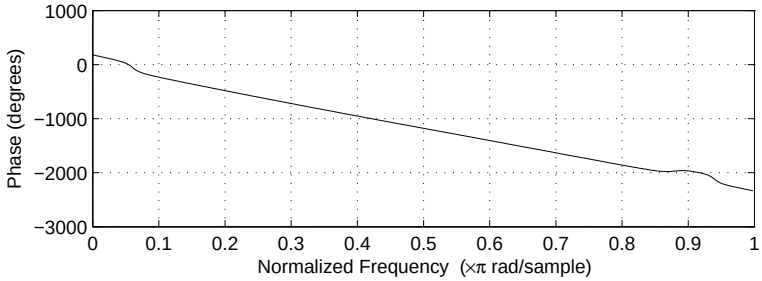

Impulse response for channel 2
h2(n)
n
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
0
5
10
15
20
25
30

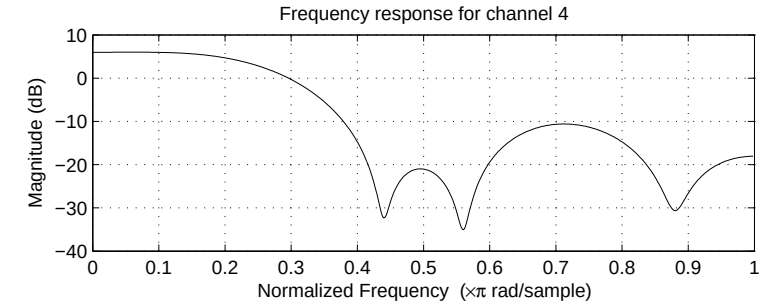

Impulse response for channel 3
h3(n)
n
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
0
5
10
15
20
25
30

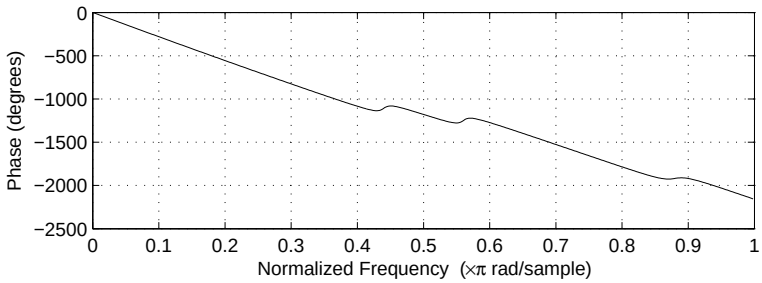

Impulse response for channel 4
h4(n)
n
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
0
5
10
15
20
25
30


Frequency response for channel 1
Magnitude (dB)
20
0
−20
−40
−60
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
−1000
−2000
−3000
Normalized Frequency (×π rad/sample)
Frequency response for channel 2
Magnitude (dB)
10
0
−10
−20
−30
−40
Normalized Frequency (×π rad/sample)
Phase (degrees)
1000
0
−1000
−2000
−3000
Normalized Frequency (×π rad/sample)









Oppgave 2 b)

The orthogonality is tested.

First, we test the orthogonality of each channel against itself. As earlier, the requirement is that the channel is orthogonal for even offsets greater than 0.

```
[R, tau] = xcorr(imp);
center=find(tau == 0);
autocol=5*(0:3)+1;

autotestrow=sort([center+(4:4:center-1), center-(4:4:center-1)]);

max(max(abs(R(autotestrow, autocol))))
```

```
ans =
      7.85046229341888e-17
```

We can see the maximal correlation is very small, an we attribute it to numerical error.

Next, we look at orthogonality between channels. We demand that the channels are orthogonal for offsets that are multiples of 4.

```
crosscol=setdiff(1:16, autocol);
crosstestrow=sort([center+(4:4:center-1), center-(4:4:center-1), center]);

max(max(abs(R(crosstestrow, crosscol))))
```

```
ans =
      7.09407033984422e-17
```

We observe this number to also be very small, and attribute the value to rounding error.

Problem 2 c)

We shall verify the integrity of a decoded signal.

```
defaultStream = RandStream.getDefaultStream();
reset(defaultStream);
L=163;
x=randn(4, L);
y=synN( synN(x(1,:), x(2,:), k), synN(x(3,:), x(4,:), k), k);
%[zprime]=zeros(2, 20016);
[zprime(1,:), zprime(2,:)]=anaN(y, k);
[z(1,:), z(2,:)]=anaN(zprime(1,:), k);
[z(3,:), z(4,:)]=anaN(zprime(2,:), k);
```

First, we must find the offset.

```
offset=find(xcorr(x(1,:), z(1,:))>=L/2)-L
```

```
offset =
     6
```

To test for corruption, we will subtract the offset output from the input.

```
max(max(x - z(:, 1+offset:L+offset)))
```

```
ans =
      2.22044604925031e-15
```

We observe only a very small number, and consider the signal uncorrupted.

Problem 2 d)

In order to introduce noise to match a SNR given in dB, we must first convert the value to a number using the `db2pow` function. We can then calculate the variance (power) of the transmitted signal to obtain the correct noise variance or power.

```
close all
clear zprime x y z
defaultStream = RandStream.getDefaultStream();
reset(defaultStream);
m=4;
L=4e6/m;
x=(randi(2, m, L) * 2) - 3;
k = tan(pi.*[0.38,0.4,0.46,0.49]);
y=synN( synN(x(1,:), x(2,:), k), synN(x(3,:), x(4,:), k), k);
n0=randn(size(y));
S=var(y);

SNR=db2pow(10);
N=S/SNR;
n=(N/var(n0))*n0;
ynoisy=y+n;
```

Problem 2 e)

We will determine the performance of the system by testing it for bit error rate with different SNRs.

To compute the BER curve to a given BER, we use a length great enough for the given BER to correspond to approximately 10 errors. In this case we will calculate to `BER=10^-5`, and should use a length of `L=10^6`.

The tested values for SNR were found experimentally.

```
snrval=fliplr(sort([-30:5:-10 -5:6 6.2 6.4 6.5]));
ival= 1:length(snrval);
offset=0;
for i = ival
    SNR=db2pow(snrval(i));
    N=S/SNR;
    n=N*n0;
    yn=y+n;
    [zprime(1,:), zprime(2,:)]=anaN(yn, k);
    [z(1,:), z(2,:)]=anaN(zprime(1,:), k);
    [z(3,:), z(4,:)]=anaN(zprime(2,:), k);
    if i==1
        offset=find(xcorr(x(1,:), z(1,:))>=max(xcorr(x(1,:), z(1,:))), 1)-L;
    end

    zregen=zeros(size(z))-1;
    idx=find(z>0);
    zregen(idx)=1;

    ierr=find(not(x == zregen(:, offset+1:L+offset)));
    nerr=length(ierr);
    BER(i)=nerr/(L*m);
end

sortrows([snrval; BER]')
```

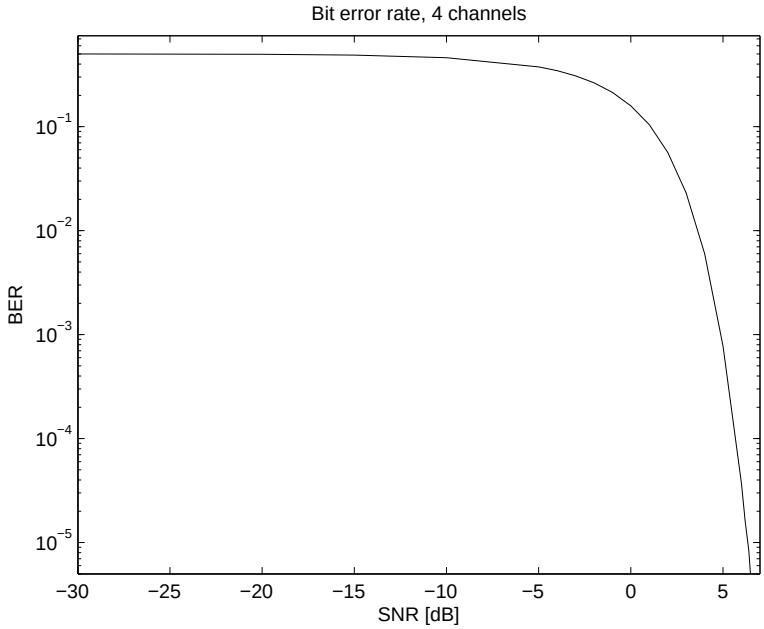
```
ans =
                       -30                 0.500107
                       -25               0.49923875
                       -20                0.4964915
                       -15                0.4878995
                       -10                 0.460656
                        -5               0.37618175
                        -4               0.34545575
                        -3                0.3083785
                        -2               0.26425975
                        -1                  0.21373
                         0                 0.158847
                         1                0.1043555
```

2	0.05647125
3	0.023068
4	0.00600125
5	0.00077925
6	3.675e-05
6.2	1.625e-05
6.4	8.25e-06
6.5	4.75e-06

Problem 2 f)

The BER curve is plotted below.

```
semilogy(snrval, BER);
axis([floor(min(snrval)) ceil(max(snrval)) 1e-5*0.5 0.5*1.5]);
title('Bit error rate, 4 channels');
xlabel('SNR [dB]');
ylabel('BER');
```



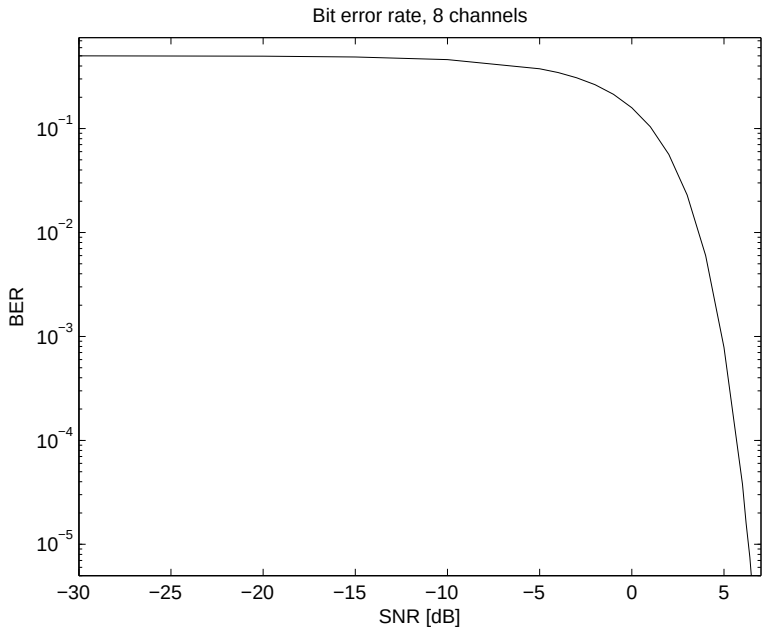
Problem 2 g)

The experiment will be repeated to determine the BER curve for a 8-channel system.

```
clear zprime z BER x y yn n0

defaultStream = RandStream.getDefaultStream();
reset(defaultStream);
m=8;
L=4e6/m;
x=(randi(2, m, L) * 2) - 3;
k = tan(pi.*[0.38,0.4,0.46,0.49]);
y=synN(synN( synN(x(1,:), x(2,:), k), synN(x(3,:), x(4,:), k), k), ...
       synN( synN(x(5,:), x(6,:), k), synN(x(7,:), x(8,:), k), k), k);
n0=randn(size(y));
S=var(y);

snrval=fliplr(sort([-30:5:-10 -5:6 6.2 6.4 6.5]));
ival= 1:length(snrval);
offset=0;
for i = ival
    SNR=db2pow(snrval(i));
    N=(S/var(n0))/SNR;
    yn=y+N*n0;
    [zprimeprime(1,:), zprimeprime(2,:)]=anaN(yn, k);

    [zprime(1,:), zprime(2,:)]=anaN(zprimeprime(1,:), k);
    [zprime(3,:), zprime(4,:)]=anaN(zprimeprime(2,:), k);
    clear zprimeprime
    [z(1,:), z(2,:)]=anaN(zprime(1,:), k);
    [z(3,:), z(4,:)]=anaN(zprime(2,:), k);
    [z(5,:), z(6,:)]=anaN(zprime(3,:), k);
    [z(7,:), z(8,:)]=anaN(zprime(4,:), k);
    clear zprime
    if i==1
        offset=find(xcorr(x(1,:), z(1,:))>=max(xcorr(x(1,:), z(1,:))), 1)-L;
    end

    zregen=zeros(size(z))-1;
    idx=find(z>0);
    zregen(idx)=1;

    ierr=find(not(x == zregen(:, offset+1:L+offset)));
    nerr=length(ierr);
    BER(i)=nerr/(L*m);
end

sortrows([snrval; BER]')

semilogy(snrval, BER);
```

```
axis([floor(min(snrval)) ceil(max(snrval)) 1e-5*0.5 0.5*1.5]);
title('Bit error rate, 8 channels');
xlabel('SNR [dB]');
ylabel('BER');
```

```
ans =
                    -30                0.4996785
                    -25               0.49880325
                    -20                 0.496114
                    -15                0.4875505
                    -10                0.4603485
                     -5               0.37598775
                     -4               0.34540675
                     -3                 0.308274
                     -2                0.2641375
                     -1               0.21361425
                      0                0.1587975
                      1               0.10394425
                      2               0.05644975
                      3               0.02296175
                      4               0.00599625
                      5                0.0007845
                      6                3.775e-05
                    6.2                  1.6e-05
                    6.4                  7.5e-06
                    6.5                  4.5e-06
```



Problem 2 h)

We find the BER curve to be close to or equal for 4 and 8 channels. We conclude that the increase of throughput only affects bandwidth.

The given building blocks could be incorporated into a system between the forward error correcting coder and the modulator. (Vice versa on the reciever side)

```
snapnow;
close all;
```