Laboratory Report

Introduction

The communication laboratory considers a digital communication system which can be illustrated as in figure 1 below.

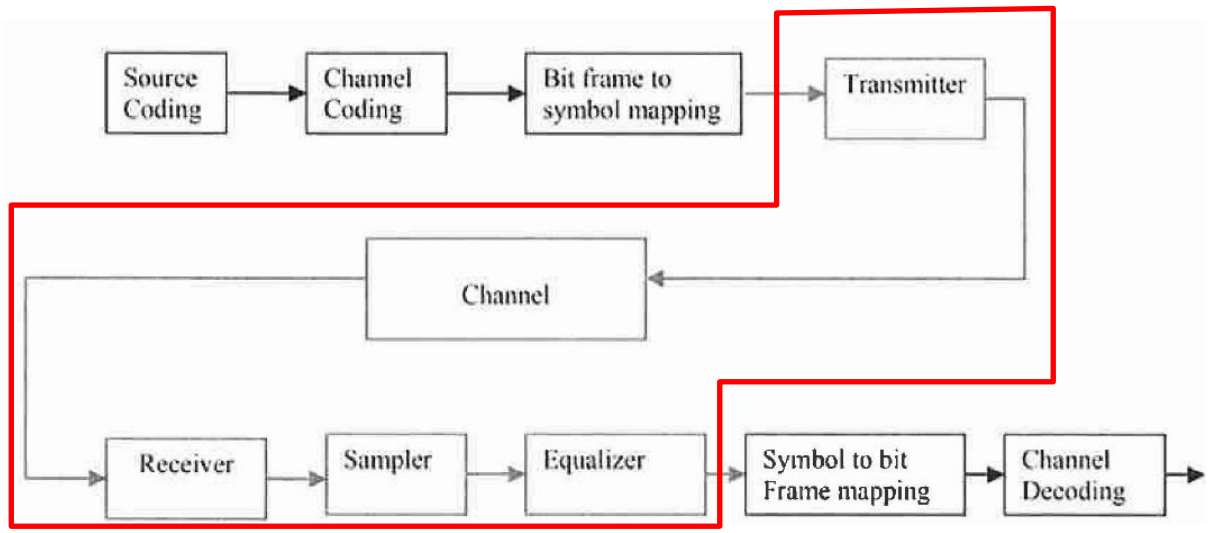

Figure 1 Digital Communication System block diagram and
the part considered in the report (red line)

However, the tasks covered in this report will only deal with the part from the transmitter to the equalizer as shown in red box in figure 1.

A time varying characteristic of a channel can be modelled by estimation for the period that the channel is flat fading, where it is constant for a period of time, in the presence of channel distortion (ISI, Inter-symbol Interference) at the receiver and noise (AWGN, Additive White Gausian Noise). The character of the channel itself is characterised by the channel impulse response, c(n), and an additive noise. An equalizer is used to compensate for the ISI by inverting the channel characteristics without enhancing the additive noise. Two types of equalization in a discrete-time channel model is shown in figure 2 and figure 3.

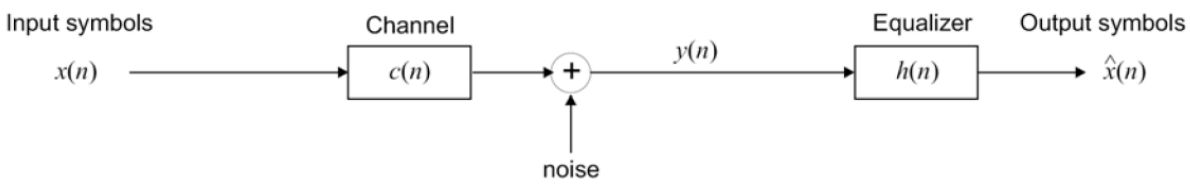

Figure 2 A symbol-spaced equalizer in a discrete-time channel model

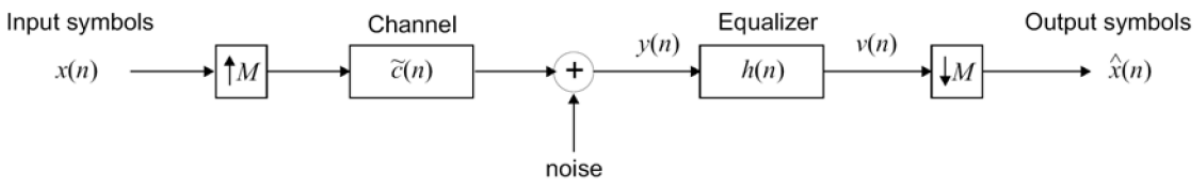

Figure 3 A fractionally-spaced equalizer in a discrete-time channel model

Laboratory Tasks

Task 1

Choose an impulse response $\tilde{c}(n)$ (for example, $\tilde{c}(n)$ = [0.04, -0.05, 0.07, -0.21, 0.5, 0.72, 0.36, 0.21, 0.03, 0.07]) . Consider M=2. To obtain c(n), take every second value of $\tilde{c}(n)$, that is c(n) = $\tilde{c}(2n)$. Then normalize c(n) to unit energy (i.e. $\sum_n |c(n)|^2 = 1$). Consider the input sequence x(n) = [1 1 -1 1 -1 1 1 -1 0 0 0 0]. Using the `stem` *command, plot x(n), y(n), $\hat{x}(n)$ for system in figure 2 and figure 3 assuming $h(n) = \delta(n)$ and no noise.*

The values of x(n), y(n) and $\hat{x}(n)$ can be calculated using the Matlab code `Task1_1` in the appendix section. In order to find the normalized value of c(n), the function `normr()` from Matlab is used. `normr(M)` is a function in Matlab that normalizes row of matrix M to the length of 1. This is done in accordance with the data in task 1, i.e. $\sum_n |c(n)|^2 = 1$.

The plotting result using `stem` function can be seen below in figure 4 and figure 5 respectively for both systems, symbol-spaced equalizer and fractionally-spaced equalizer.

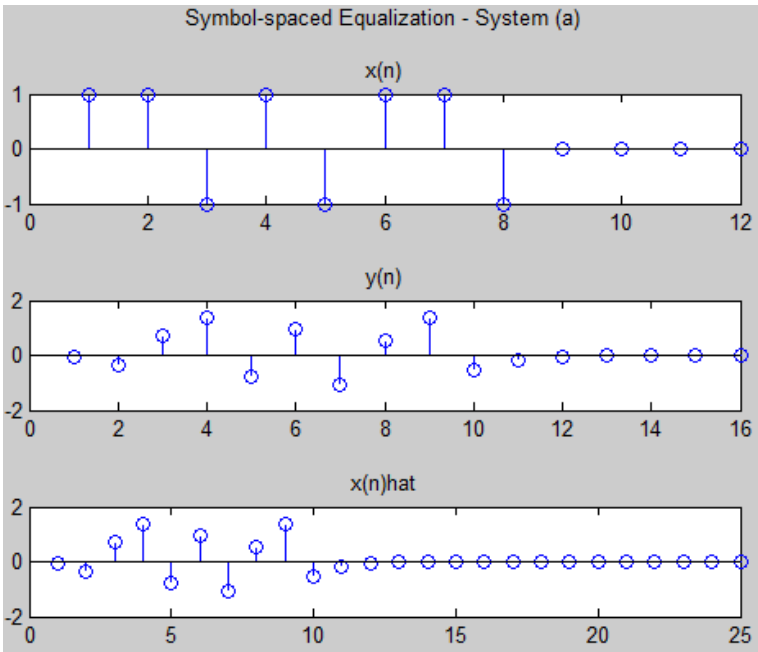

Figure 4 Matlab plots of x(n), y(n) and $\hat{x}(n)$ using symbol-spaced equalizer

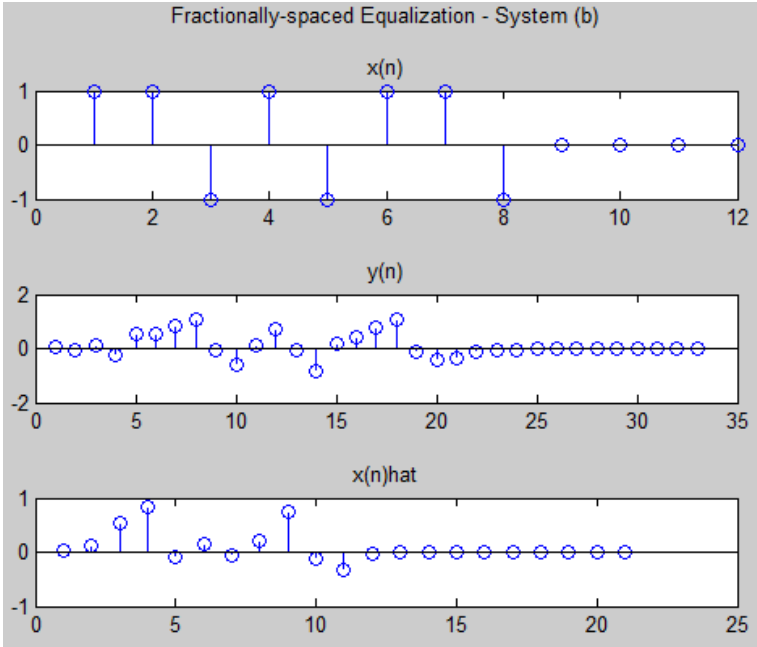

Figure 5 Matlab plots of x(n), y(n) and $\hat{x}(n)$ using fractionally-spaced equalizer

Task 2

Write Matlab functions for the equalizer design according to the ZF and MMSE criteria for ***symbol rate equalisation****. The channel impulse response c(n), the equalizer order p, the delay n0, and the signal-to-noise ratio* $SNR = 10\ log_{10}\left(\frac{\sigma_x{}^2}{\sigma_n{}^2}\right)$ *(dB) should be the input parameters.*

For symbol-rate equalization, the ZF (Zero Forcing) equalization tries to invert the channel (channel impulse response) without looking at the effects of noise which will significantly enhance noise. The aim is to find h(n) such that the convolution of c(n) by h(n) is approximately equal to zero, or $c(n) * h(n) \approx \partial(n - n0)$. By minimizing the squared error, the equalizer impulse response, h_{zf} can be calculated using the equation: $h_{zf} = [C^H C]^{-1} C^H f$, where C is c(n) matrix, f is $\partial(n - n0)$ matrix, and H denotes the transposition.

On the other hand, the MMSE (Minimum Mean Squared Error) minimizes the squared error between input and output under the influence of additive noise. The aim is to find h(n) such that the error mentioned is minimised, or $E\{|\hat{x}(n) - x(n - n0)|^2\}$ is minimised. Therefore, the solution can be written as: $h_{MMSE} = \left[C^H C + \frac{\sigma_n{}^2}{\sigma_x{}^2} I\right]^{-1} C^H f$, where σ_x is the variance of the input symbols and I is an identity matrix. From the SNR equation above, the ratio of $\frac{\sigma_n{}^2}{\sigma_x{}^2}$ can be obtain by $\frac{1}{10^{\left(\frac{SNR}{10}\right)}}$. This equation will be used in the Matlab code in this task.

Based on the underlying theories described above, the Matlab code can be structured as follows:

```
%function [h_zf h_mmse]=equalizer(cn,p,n0,SNR)

%Calculating the ratio of sigma n / sigma x for MMSE
sigma_ratio=1/(10^(SNR/10));

%Matrix C
```

```
m=length(cn_norm);
C=zeros(m+p+1,p+1); %Initialization
for i=1:p+1;
    for j=1:m;
    C(j+i-1,i)=cn_norm(j);
    end
end

%Matrix f
f=zeros(m+p+1,1);
n0=round((m+p+1)/2);    %the best delay is halfb the order
f(n0+1)=1;

%Calculate Equalizer impulse response for ZF
h_ZF=(C'*C)^(-1)*C'*f;

%Calculate Equalizer impulse response for MMSE
h_MMSE=(C'*C+e*eye(p+1))^(-1)*C'*f;
```

Task 3

Design equalisers of order 32 according to the ZF and MMSE principles for a signal-to-noise ratio SNR of 15 dB. Choose n0 such that the largest impulse lies in the middle of the overall impulse response c(n) h(n) . For both solutions plot the overall impulse responses. Compare both designs regarding the power of ISI and the power of noise at the equaliser output.*

For the purpose of this task, the result obtained in task 2 (h_ZF and h_MMSE) will be used for simulation. Therefore the value of c(n) will be taken from task 1 which is calculated from the value of $\tilde{c}(n)$. This is all done by running the Matlab code `Task1_1` in the appendix. The codes for running task 3 itself, `Task3_1` and `Task3_2`, can be seen in the appendix.

The delay chosen in this task is half the order of channel impulse response matrix C. This is in accordance to the laboratory notes that the best delay should be chosen is half of the order. Apart from that, this delay will result in the largest impulse in the middle of the convolution of c(n)*h(n).

The plots of the overall impulse can be seen in figure 6 below.

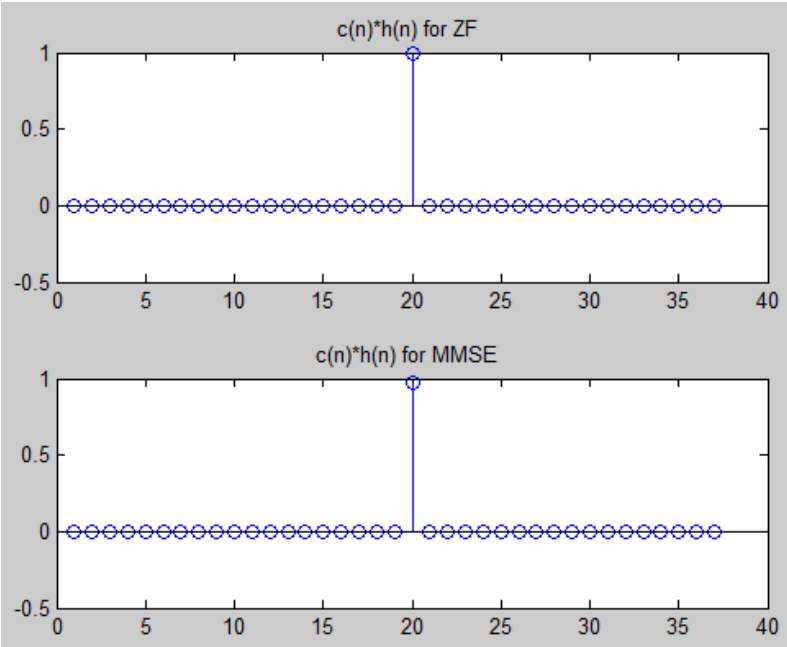

Figure 6 Matlab plots of overall impulse response c(n)*h(n) using ZF and MMSE

The power of ISI between the two approaches can be calculated using the Matlab code `Task3_2` in the appendix. The results are that **0.9679** for ZF and **0.9102** for MMSE. These results conform to the fact that ZF inverts the channel (channel impulse response) without looking at the effects of noise, while MMSE minimizes the squared error between input and output under the influence of additive noise.

Conclusion

In this lab, a discrete-time channel model is used in order to design equalizer with two different approach, the symbol rate equalization and the fractionally-spaced equalization. These two approaches is also divided into two ways in terms of the inclusion of noise, the zero forcing (ZF) and the Minimum Mean Squared Error (MMSE).

In terms of noise, it can be seen that the less noise at the output end is the better. Therefore, for the equalizer design using symbol rate equalization, the MMSE approach is better than ZF. This result is confirmed in Task 3.

However, during the lab session, it is known that it only shows a small difference in calculation results between symbol rate equalization and the fractionally-spaced equalization. Nevertheless, the fractionally-spaced equalization is better than symbol rate equalization in terms of handling ISI.

Reference list

H. Du, *ECTE955 Laboratory Notes*, University of Wollongong, 2011

J. G. Proakis, *Digital Communications*, McGraw-Hill, 4th edition.

Appendix

Matlab codes

Task1_1

```
xn=[1 1 -1 1 -1 1 1 -1 0 0 0 0];
M=2;
%Equalizer impulse response
hn=[1;zeros(9,1)];

%Channel impulse response of fractionally-spaced equalization
cn_=[0.04 -0.05 0.07 -0.21 0.5 0.72 0.36 0.21 0.03 0.07];

%Channel impulse response of symbol-spaced equalization
cn=cn_(2:M:end);
cn_norm=normr(cn);

%Symbol-spaced equalization
yn_s=conv(xn,cn_norm);
xnhat_s=conv(yn_s,hn);

%Fractionally-spaced equalization
xn_f=upsample(xn,2);
yn_f=conv(xn_f,cn_);
vn=conv(yn_f,hn);
xnhat_f=downsample(vn,2)

%Plots
%Symbol-spaced Equalization - System a
subplot(3,1,1); stem(xn); title({'Symbol-spaced Equalization - System
(a)';'x(n)'});
subplot(3,1,2); stem(yn_s); title('y(n)');
subplot(3,1,3);stem(xnhat_s); title('x(n)_hat');

%Fractionally-spaced Equalization - System b
figure;
subplot(3,1,1); stem(xn); title({'Fractionally-spaced Equalization - System
(b)';'x(n)'});
subplot(3,1,2); stem(yn_f); title('y(n)');
subplot(3,1,3); stem(xnhat_f); title('x(n)_hat');
```

Task3_1

```
ZF_all=conv(h_ZF,cn_norm);
MMSE_all=conv(h_MMSE,cn_norm);
subplot(2,1,1);
stem(ZF_all); title('c(n)*h(n) for ZF ');
subplot(2,1,2);
stem(MMSE_all); title('c(n)*h(n) for MMSE ');
```

Task3_2

```
MMSE_all(n0)=0;
ZF_all(n0)=0;
ISI_ZF=sum(ZF_all.^2);
ISI_MMSE=sum(MMSE_all.^2);
```