



EE379 - Digital Communication: Signal Processing

Extra Credit Project

Author: Dimitrios Bisias

SUID: 05336993

Introduction

This project includes:

- a) A proof of the fact that, when precoding the MAP detection is not equivalent to the ML detection
- b) A Matlab program for Canonical Factorization
- c) A different solution of 3.34
- d) A Matlab program for the Frequency Response of ZFE, MMSE-LE

A. Proof

The channel outputs when you use precoding are not equiprobable. Nevertheless, one uses ML detection. Below is the proof that this is not equivalent to MAP detection and so it is not optimum for the case $H(D) = 1-D$.

$$P(\text{error}) = 0.5 * P(\text{error}/x_k = 1) + 0.5 P(\text{error}/x_k = -1)$$

$$P(\text{error}/ x_k = 1) = 0.5P(\text{error}/ x_k = 1 \text{ and } \bar{x}_{k-1} = -1) + 0.5 P(\text{error}/ x_k = 1 \text{ and } \bar{x}_{k-1} = 1)$$

Since $\bar{m}_k = m_k \oplus \bar{m}_{k-1}$, we have:

$$P(\text{error}/ x_k = 1) = 0.5P(\text{error}/ \bar{x}_k = 1 \text{ and } \bar{x}_{k-1} = -1) + 0.5 P(\text{error}/ \bar{x}_k = -1 \text{ and } \bar{x}_{k-1} = 1)$$

$$P(\text{error}/ x_k = 1) = 0.5P(\text{error}/ y_k = 2) + 0.5 P(\text{error}/ y_k = -2) = Q\left(\frac{d - d_1}{\sigma}\right),$$

Where d_1 is the distance of the edge of the decision region from point $y = 0$.

Similarly,

$$P(\text{error}/ x_k = -1) = 0.5P(\text{error}/ x_k = -1 \text{ and } \bar{x}_{k-1} = -1) + 0.5 P(\text{error}/ x_k = -1 \text{ and } \bar{x}_{k-1} = 1)$$

$$\begin{aligned} P(\text{error}/ x_k = -1) &= 0.5P(\text{error}/ \bar{x}_k = -1 \text{ and } \bar{x}_{k-1} = -1) + 0.5 P(\text{error}/ \bar{x}_k = 1 \text{ and } \bar{x}_{k-1} = 1) \\ &= 0.5P(\text{error}/ y_k = 0, \bar{x}_{k-1} = -1) + 0.5 P(\text{error}/ y_k = 0, \bar{x}_{k-1} = 1) \end{aligned}$$

$$= 0.5P(\text{error}/ y_k = 0) = Q\left(\frac{d_1}{\sigma}\right)$$

Therefore, MAP rule is equivalent to ML rule, and so the detector is optimum.

B. Canonical Factorization

Below is a Matlab program that computes the gamma0, G(D) and G*(D-*) for a symmetric polynomial.

```
% Stanford University
% Canonical Factorization of a symmetric polynomial
% Author: Dimitrios Bisias

clear all;
% Enter the polynomial e.g [-0.9 1.81 0.0] for 0.9 D + 1.81 + 0.9D^(-1)
p = input(' Type in the polynomial coefficients (in descending factors)');

% Check if it is symmetric polynomial

check = p - conj(fliplr(p));
if( sum( check.^2 ) ~=0 )
    error(' The polynomial must be symmetric');
end

r = roots(p);

j = 0;
for (i=1:length(r))
    if(abs(r(i)) < 1)
        j = j+1;
        r_in(j) = r(i);
    end
end

G_conj = poly(r_in);
G = conj(G_conj);
gamma0 = p(1)* prod(-1./conj(r_in));

gamma0
G
G_conj
```

C. Matlab Program for 3.34

Below is a Matlab program that uses the frequency domain instead of time domain. Also there is a function for the raised cosine pulses.

```

function [y] = xrc(f,alpha,T);
% [y]=xrc(f,alpha,T)
%           evaluates the expression Xrc(f). The parameters alpha and T
%           must also be given as inputs to the function.
if (abs(f) > ((1+alpha)/(2*T))),
    y=0;
elseif (abs(f) > ((1-alpha)/(2*T))),
    y=(T/2)*(1-sin(T/(2*alpha)*(abs(2*pi*f) - pi/T)));
else
    y=T;
end;

% Data of the problem
clear;
T = 1*10^(-6); % Symbol Period
fc = 6*10^(5); % Carrier Frequency
a = 0.1;          % Excess Bandwidth
N = 1024;         % Number of FFT points

f=-1/T:2/(N*T):1/T;

for i=1:length(f)
    phi(i) = sqrt(xrc(f(i),a,T));
end

H = 1/T*10^(-6)*0.3*pi*exp(-6*pi*10^(-7)*abs(f+fc));

P = H.*phi;
P_tilda = sqrt(T)*P;

% We need to divide with the sampling rate for P(exp(j*omega*t))
P_tilda_discrete = 2/T * P_tilda;

% We shift the Fourier Transform for IFFT
P_tilda_for_IFFT = [ P_tilda_discrete(N/2+1:N+1) P_tilda_discrete(2:1:N/2)];
p = ifft(P_tilda_for_IFFT,N);

p_for_dfe = [p(end-3:1:end) p(1:4)]

% Run of DFE_Color
l = 2;
Nf = 10;
Nb = 3;
delay = 8;
Ex = 10^(-3)*T/2;
noise = []*10^(-11.65) zeros(1,Nf*l-1);
snr = dfecolor(l,p_for_dfe,Nf,Nb,delay,Ex,noisy)

snr_MFB = 10*log10(Ex*p*p'/noise(1))

```

D. Below is a Matlab program for the frequency response of ZFE and MMSE-LE.

% This program is going to study the ZFE and MMSE-LE equalizers

```

% Inputs:
% pulse response
% SNR
% Output:
% W(omega)

clear all;

p = input(' Give the pulse response ');
first_tap = input(' Give the first position of the pulse response ');
SNR = input(' Give the SNR of the system ');

% Calculate q(k)

last_tap = first_tap+length(p)-1;
norm_p = p*p';
p_reverse = conj(p(end:-1:1));
q = conv(p, p_reverse)/norm_p;

num = q;
den = [zeros(1,abs(first_tap-last_tap)) 1]

omega = linspace(0,2*pi,1024);
Q = freqz(num,den,omega);

plot(omega,abs(Q));
title(' Q ');
grid;

%ZFE Equalizer

W = 1./sqrt(norm_p)*Q;

figure;
subplot(2,1,1)
plot(omega,abs(W));
xlabel('omega');
title('Zero Force Equalizer Frequency Response');
grid;

% MMSE-LE Equalizer

W_LE = 1./sqrt(norm_p)*(Q + 1/(norm(p)*SNR) );

subplot(2,1,2)
plot(omega,abs(W_LE));
xlabel('omega');
title('MMSE-LE Equalizer Frequency Response');
grid;

```