Professor Fitz
3/9/09

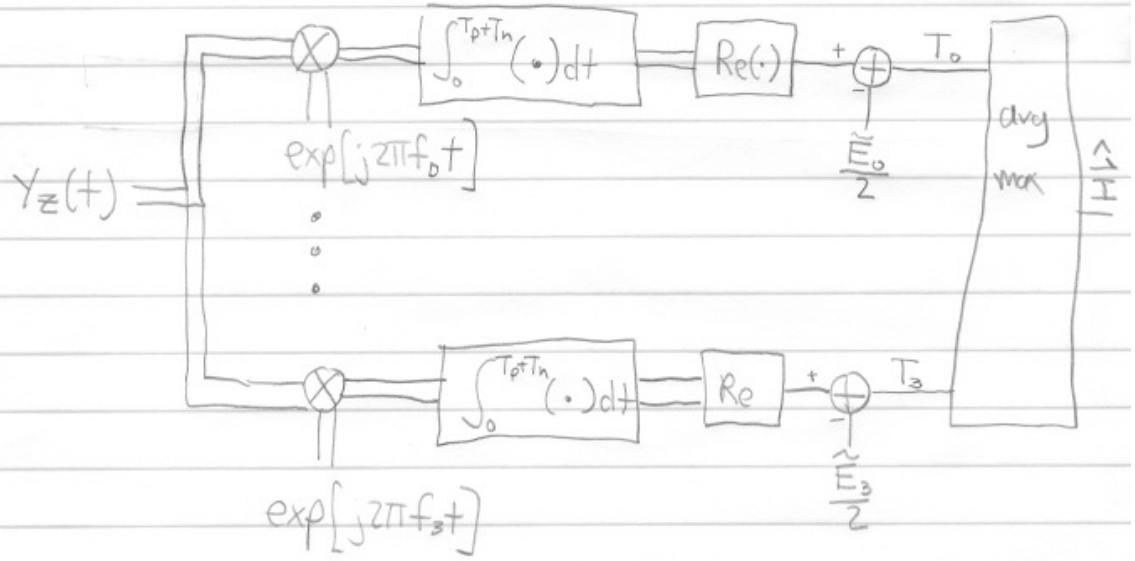
Problem fs. 1

$$r_i(t) = \int_0^{T_h} h_r(\lambda)\, X_i(t-\lambda)\, d\lambda$$

$$= \int_0^{T_p/4} \left(\alpha_0 \delta(\lambda) + \alpha_1 \delta\left(\lambda - \frac{T}{4}\right) \right) X_i(t)$$

$$= \alpha_0 X_i(t) + \alpha_1 X_i\left(t - \frac{T}{4}\right)$$

a) Optimal demodulator



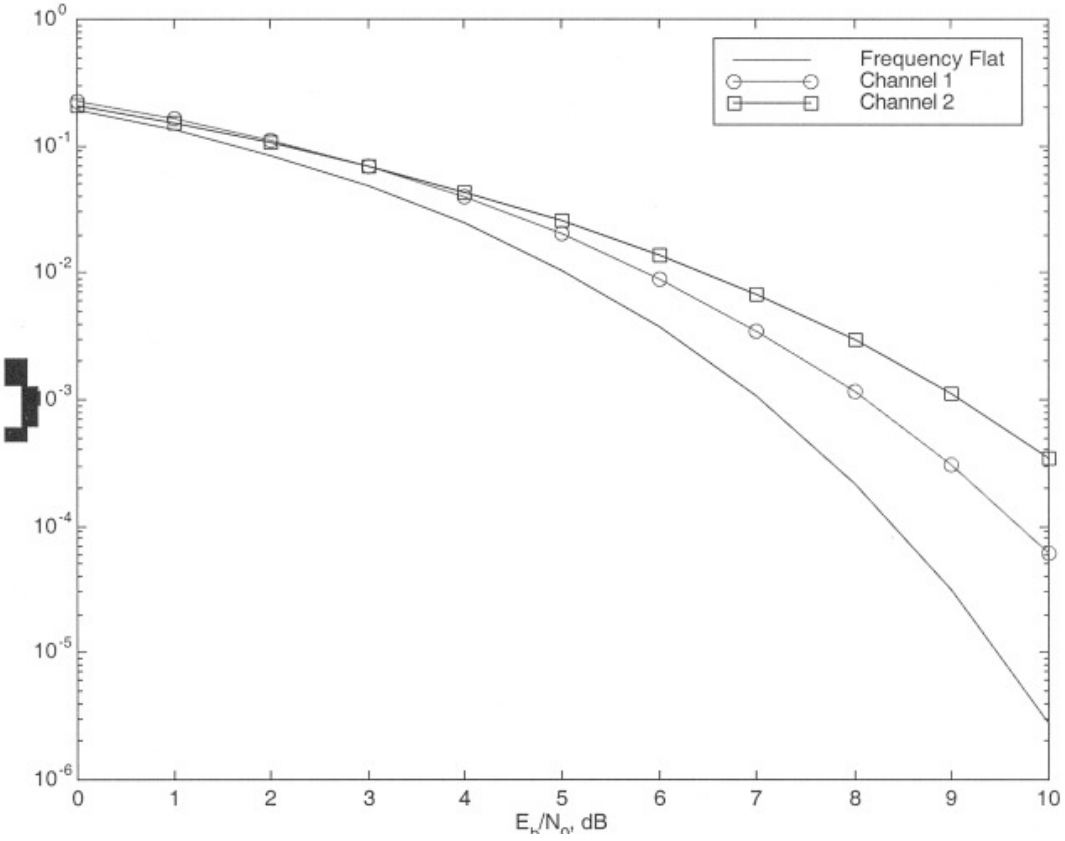
$$\tilde{E}_i = \int_0^{T_p+T_h} |r_i(t)|^2 dt = E_i + \alpha_0 \alpha_i^* R_{x_i}\left(\frac{T}{4}\right) + \alpha_1 \alpha_0^* R_{x_i}\left(-\frac{T}{4}\right)$$

$$E_b = \frac{\sum_{i=0}^{M-1} \tilde{E}_i}{K_b \cdot M} = \frac{\sum_{i=0}^{M-1} \tilde{E}_i}{8}$$

b) $$P_{WUB}(E) = \sum_{i=0}^{3} \sum_{i \neq j} \frac{1}{8} \operatorname{erfc}\left[\sqrt{\frac{\Delta_E(i,j)}{4N_0}}\right]$$

$$\Delta_{\tilde{E}}(i,j) = \int_0^{T_p+T_n} \left| \alpha_0\left(x_i(t) - x_j(t)\right) + \alpha_1\left(x_i(t) - x_j(t)\right) \right|^2 dt$$

Note: The optimum frequency spacing is computed in Chapter "Move bits" Problem 10.


Frequency Flat
Channel 1
Channel 2
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
1
2
3
4
5
6
7
8
9
10
E_b/N_0, dB

```
% Digital Communication Theory
%
% Computing the union bound for M-ary FSK
% as a function of frequency deviation.
%
% Author: M. Fitz
% Last Revision: 3/9/04
%
numpts=1001;
numptsr=1251;
timetx=linspace(0,1,numpts);
timerx=linspace(0,1.25,numptsr);
%
% a) Computing E_b
%
%
% Optimum f_d
%
f_d=0.429;
%f_d=0.25;
%
% Transmitted signals
%
x_z(1,:)=exp(-j*2*pi*3*f_d*timetx);
x_z(2,:)=exp(-j*2*pi*f_d*timetx);
x_z(3,:)=exp(j*2*pi*f_d*timetx);
x_z(4,:)=exp(j*2*pi*3*f_d*timetx);
%
%  Channel stuff
%
alpha_11=0.3*exp(j*2*pi/3);
alpha_12=0.7*exp(j*pi/3);
alpha_01=sqrt(1-abs(alpha_11)^2);
alpha_02=sqrt(1-abs(alpha_12)^2);
%
%  Computing the received signal : Frequency Flat
%
r_0=zeros(4,1251);
for  ii=1:4
    r_0(ii,1:numpts)=x_z(ii,:);
end
E_tilde_0=diag(real(r_0*r_0'/numpts))
E_b0=sum(E_tilde_0)/8
%
%  Computing the received signal : Channel 1
%
r_1=zeros(4,1251);
for  ii=1:4
    r_1(ii,1:numpts)=alpha_01*x_z(ii,:);
    r_1(ii,251:250+numpts)=r_1(ii,251:250+numpts)+alpha_11*x_z(ii,:);
end
E_tilde_1=diag(real(r_1*r_1'/numpts))
E_b1=sum(E_tilde_1)/8
%
%  Computing the received signal : Channel 2
%
r_2=zeros(4,1251);
for  ii=1:4
    r_2(ii,1:numpts)=alpha_02*x_z(ii,:);
    r_2(ii,251:250+numpts)=r_2(ii,251:250+numpts)+alpha_12*x_z(ii,:);
end
E_tilde_2=diag(real(r_2*r_2'/numpts))
```

```
E_b2=sum(E_tilde_2)/8
%
% b) Computing Euclidean distance
%
%
%  Computing the Euclidean distance : Frequency flat
%
for jj=1:4
    for ii=1:4
        delta=r_0(ii,:)-r_0(jj,:);
        Delta_E_0(ii,jj)=sum(abs(delta).^2)/E_b1/numpts;
    end
end
Delta_E_0
%
%  Computing the Euclidean distance : Channel 1
%
for jj=1:4
    for ii=1:4
        delta=r_1(ii,:)-r_1(jj,:);
        Delta_E_1(ii,jj)=sum(abs(delta).^2)/E_b1/numpts;
    end
end
Delta_E_1
%
%  Computing the Euclidean distance : Channel 2
%
for jj=1:4
    for ii=1:4
        delta=r_2(ii,:)-r_2(jj,:);
        Delta_E_2(ii,jj)=sum(abs(delta).^2)/E_b1/numpts;
    end
end
Delta_E_2
%
%  Plotting the performance
%
figure(1)
snr_db=(0:10);
snr=10.^(snr_db/10);
P_WUB=zeros(3,11);
for jj=1:4
    for ii=1:4
        if(ii==jj)
        else
            P_WUB(1,:)=P_WUB(1,:)+0.125*erfc(sqrt(snr*Delta_E_0(ii,jj)/4));
            P_WUB(2,:)=P_WUB(2,:)+0.125*erfc(sqrt(snr*Delta_E_1(ii,jj)/4));
            P_WUB(3,:)=P_WUB(3,:)+0.125*erfc(sqrt(snr*Delta_E_2(ii,jj)/4));
        end
    end
end
semilogy(snr_db,P_WUB(1,:),snr_db,P_WUB(2,:),'r-o',snr_db,P_WUB(3,:),'ks-')
xlabel('E_b/N_0, dB')
ylabel('P_{WUB}(E)')
legend('Frequency Flat','Channel 1','Channel 2')
```