ADAPTIVE MODULATION

clc; clear; close all;

s=randi([0,1],1,1000);

error=[];

op=[];

v=1/length(s);

for snr=1:30

if snr>=0 && snr<=13

x=pskmod(s,2)

end

if snr>=13 && snr<=20

x=pskmod(s,4)

end

if snr>=20 && snr<=26

x=qammod(s,16)

end

if snr>=26 && snr<=30

x=qammod(s,64)

end

y=awgn(x,snr)

if snr>=0 && snr<=13

dm=pskdemod(y,2)

end

if snr>=13 && snr<=20

dm=pskdemod(y,4)

end

if snr>=20 && snr<=26

dm=qamdemod(y,16)

end

if snr>=26 && snr<=30

dm=qamdemod (y,64)

end

[num,rat]=biterr(s,dm);

if rat==0

rat=v;

end

error=[error rat];

end

subplot(2,1,1)

semilogy(0:29,error)

throughput=[];

for i =1:30

if i>=0 && i<13

throughput= [throughput 1];

disp(throughput);

end;

if i>=13 && i<20

throughput= [throughput 2];

disp(throughput);

end;

if i>=20 && i<26

throughput= [throughput 4];

disp(throughput);

end;

if i>=26 && i<=30

throughput= [throughput 6];

disp(throughput);

end

end

subplot(2,1,2);

semilogy(0:29,throughput)

EQUALIZER ZFE ONLY

clc;

clear all;

close all;

n = randi([0,1],1,10000);

message=n;

h = randn(1,10000) + 1i\*randn(1,10000); % Random channel response (complex)

y = pskmod(n,2); % BPSK modulation

% Transmitted signal

sig = y.\*h;

SNRdB = 0:20; % Range of SNR values in dB

ber = zeros(size(SNRdB)); % Initialize BER vector

ber1 = zeros(size(SNRdB))

for snrIdx = 1:length(SNRdB)

snr = SNRdB(snrIdx);

% Add AWGN to the signal

noise = awgn(sig, snr);

% Equalization (Zero Forcing Equalizer)

eq = noise.\*(1./h);

% Demodulate the received signal

demod = pskdemod(eq,2);

demod1= pskdemod(noise,2);

% Calculate BER

[num, rat] = biterr(demod, n);

ber(snrIdx) = rat;

[num1, rat1] = biterr(demod1, n);

ber1(snrIdx) = rat1;

end

% Plot BER vs. SNR

semilogy(SNRdB, ber, '-k','LineWidth',1.2);

grid on;

hold on;

semilogy(SNRdB, ber1, '--k','LineWidth',1.2);

title('Bit Error Rate vs. SNR');

legend('BER with equalizer','BER without equalizer');

grid on;

EQUALIZER LMS AND ZFE

clc; clear all; close all;

message = randi([0,1],1,10000);

snr = 0:2:40; mod = 2;

L = 2; r = 3; % 3 TAP EQUALISER

modulated\_bpsk\_msg = pskmod(message,mod);

h0 = 1; h1 = 0.7;

H = zeros(r,r+L-1); % ORDER OF H MATRIX

% FORMING H MATRIX FOR 3 TAP EQUALISER

for p = 1:r

H(p,p:p+L-1) = [h0 h1];

end

x = []; % FORMING INPUT MATRIX X

X = modulated\_bpsk\_msg; X\_1 = circshift(X,1);

X\_1(1) = 0; X1 = circshift(X,-1);

X1(end) = 0; X2 = circshift(X,-2);

X2(end-1:end)= 0; x = [X2;X1;X;X\_1];

C = ((H\*H')\H)\*[0;0;1;0];

ber\_without\_ZFE = []; ber\_with\_ZFE = [];

for p = 1:length(snr)

y = awgn([h0 h1]\*[X;X\_1],snr(p),'measured');

Noise = y - ([h0 h1]\*[X;X\_1]);

Noise\_1 = circshift(Noise,-1);

Noise\_1(end) = 0; Noise\_2 = circshift(Noise,-2);

Noise\_1(end-1:end) = 0;

Y = (H\*x)+ [Noise\_2;Noise\_1;Noise]; X\_PRIME = C.'\*Y;

Demodulated\_BPSK\_msg\_with\_ZFE =pskdemod(X\_PRIME',mod);

Demodulated\_BPSK\_msg\_without\_ZFE = pskdemod(y',mod);

[number1,ratio1] = biterr(message,Demodulated\_BPSK\_msg\_with\_ZFE');

[number2,ratio2] = biterr(message,Demodulated\_BPSK\_msg\_without\_ZFE');

ber\_with\_ZFE = [ber\_with\_ZFE,ratio1];

ber\_without\_ZFE = [ber\_without\_ZFE,ratio2];

end

semilogy(snr,ber\_with\_ZFE,'-k','LineWidth',1.6)

hold on

semilogy(snr,ber\_without\_ZFE,'--k','LineWidth',1.8)

hold on

n = length(message); M = 25; w = zeros(1,M);

wi = zeros(1,M); E = []; mu = 0.0005;

msg\_bpsk = pskmod(message,mod);

ber\_without\_LMS = []; ber\_with\_LMS = [];

for k = 1:length(snr)

msg\_rx = awgn(msg\_bpsk, snr(k), 'measured');

for i = M:n

E(i) = msg\_bpsk(i) - wi\*msg\_rx(i:-1:i-M+1)';

wi = wi + 2\*mu\*E(i)\*msg\_rx(i:-1:i-M+1);

end

msg\_eq = zeros(n,1);

for i = M:n

j = msg\_rx(i:-1:i-M+1);

msg\_eq(i) = ((wi)\*(j)');

end

Demod\_with\_LMS = pskdemod(msg\_eq,mod)';

Demod\_without\_LMS = pskdemod(msg\_rx,mod);

[n1,r1] = biterr(message,Demod\_with\_LMS);

[n2,r2] = biterr(message,Demod\_without\_LMS);

ber\_without\_LMS = [ber\_without\_LMS,r1];

ber\_with\_LMS = [ber\_with\_LMS,r2];

end

grid on; semilogy(snr,ber\_with\_LMS,'-\*k','LineWidth',1.2)

title('SNR VS BER'); xlabel('SNR(dB)');ylabel('BER');

legend('BER WITHOUT EQUALIZATION','BER WITH ZFE','BER WITH LMS');

FADING

clc;

clear all;

close all;

fs = 4000; f1 = 1000; f2 = 400;

ts = 0:(1/fs):1-(1/fs);

signal1 = exp(complex(0,2\*pi\*f1\*ts));

signal2 = exp(complex(0,2\*pi\*f2\*ts));

message = signal1+signal2;

delayed = [zeros(1,200) message(1:length(message)-200)];

%FREQUENCY RESPONSE

freqres = fft(message); figure;

magnitudeplot = abs(freqres);

plot(1:fs,magnitudeplot);

title({'Input Signal';'Frequency Response'});

xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');

figure; freqres2=fft(delayed);

plot(1:fs,abs(freqres2));

title({'Delayed Input signal';'Frequency Response'});

xlabel('Frequency(Hz)');

ylabel('Amplitude(V)');

%Slow and Flat Fading

h = randn + (1i\*randn);

y1 = message.\*h;

freqres = fft(y1);

magnitudeplot = abs(fft(y1));

figure;

plot(1:fs,magnitudeplot(1:fs));

title('Slow and Flat Fading');

xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');

%Slow and Frequency Selective Fading

h1 = randn + (1i\*randn);

h2 = randn + (1i\*randn);

trans2 = (h1.\*message) + (h2.\*delayed);

magnitudeplot = abs(fft(trans2));

figure; plot(1:fs,magnitudeplot);

title('Slow and Frequency Selective Fading');

xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');

%Fast and Flat Fading

fd = 10;%doppler frequency

hs = randi([0 1],1,length(ts)) +(1i\*randi([0 1],1,length(ts)));

[b,a] = butter(12,((2\*fd)/1000));

lpf = filter(b,a,hs);

trans3 = lpf.\*message;

magnitudeplot = abs(fft(trans3));

figure; plot(1:fs,magnitudeplot);

title('Fast and Flat Fading');

xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');

% Fast and Frequency Selective Fading

ht = randi([0 1],1,length(ts)) +1i\*(randi([0 1],1,length(ts)));

lpf1 = filter(b,a,ht);

ht1 = randi([0 1],1,length(ts)) +1i\*(randi([0 1],1,length(ts)));

lpf2 = filter(b,a,ht1);

trans4 = (lpf1.\*message) +(lpf2.\*delayed);

magnitudeplot = abs(fft(trans4));

figure;

plot(1:fs,magnitudeplot);

title('Fast and Frequency Selective Fading');

xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');

OFDM

clc;clear ;close all;

x=randi([0,1],1,4096);

snr=0:30;

biterror=[];

for i=1:4

if(i==1)

y=pskmod(x,2^i);

end

if(i==2)

y=pskmod(x,2^i);

end

if(i==3)

y=qammod(x,2^(i+1));

end

if(i==4)

y=qammod(x,2^(i+2));

end

p=reshape(y,64,64);

q=ifft(p,64);

s=reshape(q,1,4096);

be=[];

for j=0:1:30

h=randn+i\*randn

r=h\*s;

n=awgn(r,j,'measured');

m=inv(h)\*n;

p1=reshape(m,64,64);

q1=fft(p1,64);

s1=reshape(q1,1,4096);

if(i==1)

y1=pskdemod(s1,2);

end

if(i==2)

y1=pskdemod(s1,4);

end

if(i==3)

y1=qamdemod(s1,16);

end

if(i==4)

y1=qamdemod(s1,64);

end

[num,e]=symerr(y1,x);

be=[be e];

end

biterror(i,:)=be;

end

semilogy(snr,biterror(1,:),'k');hold on;

semilogy(snr,biterror(2,:),'m');hold on;

semilogy(snr,biterror(3,:),'r');hold on;

semilogy(snr,biterror(4,:),'b');hold on;

xlabel('SNR(dB)');ylabel('BER');

title('SNR VS BER');legend('BPSK','QPSK','16QAM','64QAM');

RC4

clc;clear all;close all;

pt1=input("Enter your message to be encrypted : ",'s');

pt=double(char(pt1));

key1=input("Enter your key : ",'s');

key=double(char(key1));

s=0:255;

t1=[];

for ii=1:256

if length(t1)<=256

z=t1;

t1=[key z];

else

break

end

end

t=t1(1:256);

j=0;

for ii=1:256

j=mod(j+s(ii)+t(mod(ii,length(t))+1),256)+1;

a=s(ii);

s(j)=s(ii); s(ii)=a;

end

j=0;

kstr1=0;

kstr=[];

for ii=1:length(pt)

i=mod((ii+1),256);

j=mod((j+s(i)),256);

kstr1=s(mod((s(i)+s(j)),256));

kstr=[kstr kstr1];

end

keystream=kstr;

ct=[];

ptt=[];

for ii=1:length(keystream)

ct1=bitxor(pt(ii),keystream(ii));

ct=[ct ct1];

ptt1=bitxor(ct(ii),keystream(ii));

ptt=[ptt ptt1];

end

Ciphertext=char(ct)

Plaintext=char(ptt)

STBC

clc; clear all; close all;

n = randi([0,1],1,4096);

a = reshape(n,length(n),1);

bpskmod = pskmod(a,2);

snr = 0:1:30; h1 = 2+1j; h2 = 1-2j;

y = []; M= []; Op = [];

OpwoutSTBC = []; q = 1;

for l = 1:length(snr)

for p = 1:2:length((n))-1

c1 = (h1\*bpskmod(p,1))+(h2\*bpskmod(p+1,1));

M(p,q) = awgn(c1,snr(l),'measured');

c2 = (-h1\*conj(bpskmod(p+1,1)))+(h2\*conj(bpskmod(p,1)));

M(p+1,q) = awgn(c2,snr(l),'measured');

end

for r = 1:2:(length(n) -1 )

y(r,q) = (conj(h1)\*M(r,1)) + h2\*conj(M(r+1,1));

y(r+1,q) = (conj(h2)\*M(r,1)) - h1\*conj(M(r+1,1));

end

t1 = pskdemod(y,2);

Op(l,:) = reshape (t1,1,length(n));

[number,ratio] = biterr(Op,n);

end

semilogy(snr,ratio,'pentagram--C','Color','black')

xlabel("SNR(in dB"); ylabel("BER"); hold on

%without STBC

for l = 1:length(snr)

rec = awgn((h1\*bpskmod),snr(l),"measured");

demod = pskdemod(rec,2);

OpwoutSTBC(l,:) = reshape(demod,1,length(n));

[number1,ratio1] = biterr(OpwoutSTBC,n);

end

semilogy(snr,ratio1,'-k')

grid on

legend('BER with STBC','BER without STBC')

xlabel("SNR(in dB"); ylabel("BER");

title("SNR vs BER");

WIRELESS CHANNELS

clc;

clear all;

close all;

n = 10000;

i = randi([0,1],1,n);

i1 = 2\*i - 1;

a = randn(1,n);

b = randn(1,n);

rc = 1/sqrt(2) \* (sqrt(a.^2 + b.^2));

% BER vs SNR

figure;

for l = 0:1:20

snr = 10^(l/10);

sdev = sqrt(0.5/snr);

N = random('norm',0,sdev,[1,n]);

yrc = rc .\* i1 + N;

YR = (yrc >= 0);

ErrorR = sum((xor(YR,i)));

ber\_R(l+1) = ErrorR/n;

end

q = 0:1:20;

semilogy(q, ber\_R(q+1), 'k-', 'LineWidth', 2);

hold on;

axis([0 20 10^-5 1]);

xlabel('SNR (dB)');

ylabel('BER');

k1 = 10;

mean\_Rician = sqrt(k1/(k1+1));

sigma\_Rician = sqrt(1/(2\*(k1+1)));

Nr2 = randn(1,length(i1))\*sigma\_Rician + mean\_Rician;

Ni2 = randn(1,length(i1))\*sigma\_Rician;

No3 = sqrt(Nr2.^2 + Ni2.^2);

for k = 0:1:20

snr\_Rician = 10^(k/10);

Np\_Rician = 1/snr\_Rician;

sd\_Rician = sqrt(Np\_Rician/2);

No\_Rician = random('Normal', 0, sd\_Rician, 1, length(i1));

t1 = i1.\*No3 + No\_Rician;

z1 = t1./No3;

op1\_Rician = (z1 > 0);

Berr\_Rician(k+1) = sum(xor(op1\_Rician, i))/n;

end

k\_AWGN = 0:1:20;

for k = 0:1:20

snr\_AWGN = 10^(k/10);

sdev\_AWGN = sqrt(0.5/snr\_AWGN);

N\_AWGN = random('norm', 0, sdev\_AWGN, [1, n]);

y\_AWGN = i1 + N\_AWGN;

YA = (y\_AWGN >= 0);

Error\_AWGN = sum((xor(YA, i)));

Berr\_AWGN(k+1) = Error\_AWGN/n;

end

semilogy(k\_AWGN, Berr\_AWGN(k\_AWGN+1), '^-', 'DisplayName', 'AWGN Channel');

semilogy(0:20, Berr\_Rician, '\*-', 'DisplayName', 'Rician Channel');

title('BER Performance');

xlabel('SNR (dB)');

ylabel('BER');

legend('Rayleigh Channel', 'AWGN Channel', 'Rician Channel', 'Location', 'best');

grid on;

% BER vs Channel Coefficient

figure;

k\_values = 0:0.1:2;

% Rayleigh channel

for k\_index = 1:length(k\_values)

k = k\_values(k\_index);

a = randn(1,n);

b = randn(1,n);

rc = 1/sqrt(2) \* (sqrt(a.^2 + b.^2));

y\_rc = rc .\* i1;

Y\_rc = (y\_rc >= 0);

Error\_rc = sum((xor(Y\_rc, i)));

ber\_rc(k\_index) = Error\_rc/n;

end

% Rician channel

for k\_index = 1:length(k\_values)

k = k\_values(k\_index);

mean\_Rician = sqrt(k/(k+1));

sigma\_Rician = sqrt(1/(2\*(k+1)));

Nr = randn(1, n) \* sigma\_Rician + mean\_Rician;

Ni = randn(1, n) \* sigma\_Rician;

No = sqrt(Nr.^2 + Ni.^2);

t\_rician = i1 .\* No;

z\_rician = t\_rician ./ No;

op\_rician = (z\_rician > 0);

Error\_rician = sum(xor(op\_rician, i));

ber\_rician(k\_index) = Error\_rician/n;

end

% AWGN channel

for k\_index = 1:length(k\_values)

k = k\_values(k\_index);

sdev\_AWGN = sqrt(0.5);

N\_AWGN = random('norm', 0, sdev\_AWGN, [1, n]);

y\_AWGN = i1 + N\_AWGN;

YA\_AWGN = (y\_AWGN >= 0);

Error\_AWGN = sum(xor(YA\_AWGN, i));

ber\_AWGN(k\_index) = Error\_AWGN/n;

end

semilogy(k\_values, ber\_rc, 'o-', 'DisplayName', 'Rayleigh Channel');

semilogy(k\_values, ber\_rician, 's-', 'DisplayName', 'Rician Channel');

semilogy(k\_values, ber\_AWGN, '^-', 'DisplayName', 'AWGN Channel');

xlabel('Channel Coefficient (k)');

ylabel('Bit Error Rate (BER)');

legend('Location', 'best');

title('BER vs Channel Coefficient for Rayleigh, Rician, and AWGN Channels');

grid on