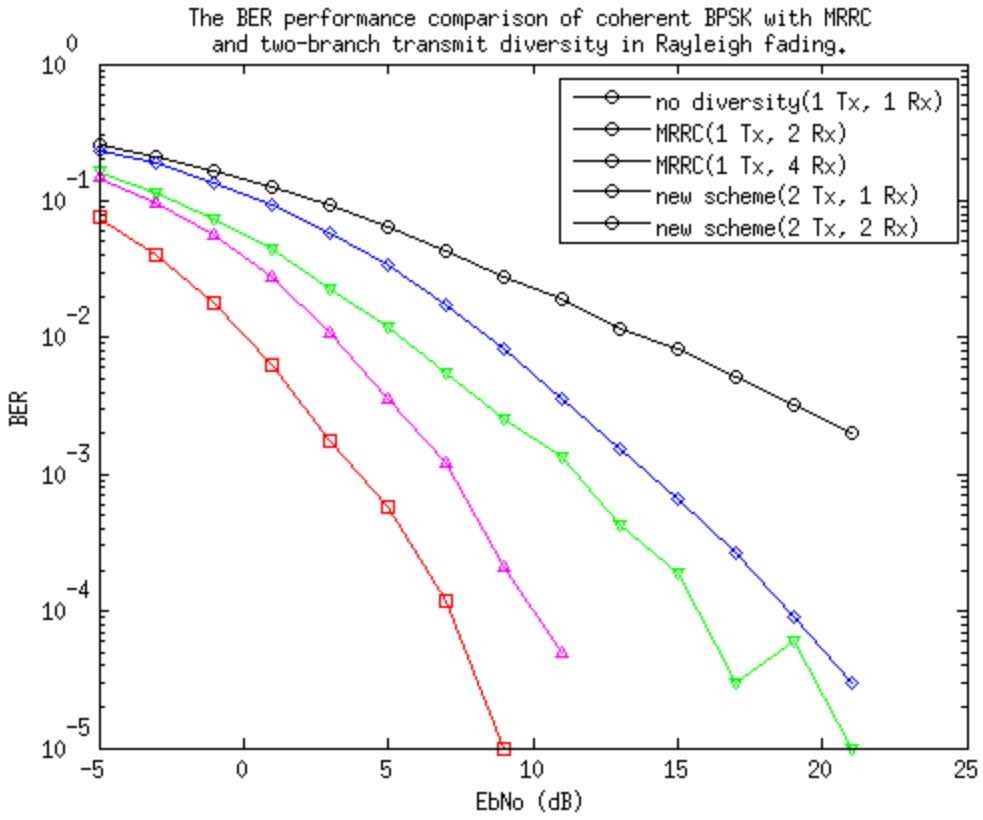
Wireless Comms mini Matlab 3

```
%Neema Aggarwal
%Shivam Mevawala
%Nicolas Castro

close all;
clear all;
SNR = -5:2:21; %list of SNR values to run algorithm
%intialize vecs
BER_reg=zeros(length(SNR));
BER_mrrc2=zeros(length(SNR));
BER_mrrc4=zeros(length(SNR));
BER_new=zeros(length(SNR));
BER_tbt=zeros(length(SNR));


n=1e5; %number of samples
m=2; %BPSK is 2-QAM

% delayVector = [0 1 2 3 4] * 1e-5; % Discrete delays of four-path channel (s)
% gainVector = [0 -4 -6 -9 -14];

rchan_flat=rayleighchan(1e-5,1e4); %Set up the channel fading object
rchan_flat.StoreHistory = 1; %If this value is 1, channel state information
%needed by the channel visualization tool is stored.
rchan_flat.StorePathGains = 1;%If set to 1, the complex path gain vector is
%stored.
rchan_flat2=rayleighchan(1e-5,1e4);
rchan_flat2.StoreHistory = 1;
rchan_flat2.StorePathGains = 1;

rchan_flat3=rayleighchan(1e-5,1e4);
rchan_flat3.StoreHistory = 1;
rchan_flat3.StorePathGains = 1;

rchan_flat4=rayleighchan(1e-5,1e4);
rchan_flat4.StoreHistory = 1;
rchan_flat4.StorePathGains = 1;

rchan_flat5=rayleighchan(1e-5,1e4);
rchan_flat5.StoreHistory = 1;
rchan_flat5.StorePathGains = 1;

rchan_flat6=rayleighchan(1e-5,1e4);
rchan_flat6.StoreHistory = 1;
rchan_flat6.StorePathGains = 1;

rchan_flat7=rayleighchan(1e-5,1e4);
rchan_flat7.StoreHistory = 1;
rchan_flat7.StorePathGains = 1;
```

```
rchan_flat8=rayleighchan(1e-5,1e4);
rchan_flat8.StoreHistory = 1;
rchan_flat8.StorePathGains = 1;


%use the SNR to calculate EbNo for the normal sytem and the convolutional
%coder
EbNo = SNR -10*log10(log2(m));


%loop over SNR values
for k=1:length(SNR)
    %generate a random vector of 4 symbols
    X=randi([0 m-1],1,n);
    %modulate
    Y=qammod(X,m);
    Y0=zeros(1,length(Y)/2);
    Y1=zeros(1,length(Y)/2);

    for kk=1:length(Y)
        if mod(kk,2) == 0 % number is even
            Y1(kk-1)=Y(kk);
            Y0(kk)=-conj(Y(kk));
        else
            %number is odd
            Y0(kk)=Y(kk);
            Y1(kk+1)=conj(Y(kk));
        end

    end

    %add noise and rayleigh fading
    A=filter(rchan_flat,Y);
    A2=filter(rchan_flat2,Y);
    A3=filter(rchan_flat3,Y);
    A4=filter(rchan_flat4,Y);

    filter(rchan_flat5,Y0); %instantiate the Path Gains
    filter(rchan_flat6,Y1);

    filter(rchan_flat7,Y0); %instantiate the Path Gains
    filter(rchan_flat8,Y1);

%     Anewc0=filter(rchan_flat6,conj(Y0));
%     Anewc1=filter(rchan_flat5,-conj(Y1));

    Ag = awgn(A, SNR(k),'measured');
    Ag2 = awgn(A2, SNR(k),'measured');
    Ag3 = awgn(A3, SNR(k),'measured');
    Ag4 = awgn(A4, SNR(k),'measured');

    h0 = rchan_flat5.PathGains.'; %Make the h vecotors match the
                                  %ones in the paper
    h1 = rchan_flat6.PathGains.';
```

```
h2 = rchan_flat7.PathGains.';
h3 = rchan_flat8.PathGains.';

for kkk=1:(length(Y)/2)
    h0(2*kkk) = h0(2*kkk-1); %Set the gains of the channel such that
                             %they don't change between sending s0 and s1
    h1(2*kkk) = h1(2*kkk-1);
    h2(2*kkk) = h2(2*kkk-1);
    h3(2*kkk) = h3(2*kkk-1);
end

%place holder variables
Anew0=h0.*Y0;
Anew1=h1.*Y1;
Atbt0 = h0.*Y0;
Atbt1 = h1.*Y1;
Atbt2 = h2.*Y0;
Atbt3 = h3.*Y1;

%Add noise
Rnew=awgn(Anew0+Anew1,SNR(k),'measured');
Dnew=zeros(1,length(Y));

%Add noise
Rtbt0 = awgn(Atbt0+Atbt1,SNR(k),'measured');
Rtbt1 = awgn(Atbt2+Atbt3,SNR(k),'measured');
Dtbt = zeros(1,length(Y));

for kkk=1:(length(Y)/2) %Combiner
    %split Rnew into vectors that match the equations in the paper
    r0=Rnew(2*kkk-1);
    r1=Rnew(2*kkk);

    %Combine
    Dnew(2*kkk-1)= conj(h0(2*kkk-1))*r0 + h1(2*kkk)*conj(r1);
    Dnew(2*kkk)= conj(h1(2*kkk-1))*r0-h0(2*kkk)*conj(r1);

    %split Rtbt into vectors that match the equations in the paper
    r0=Rtbt0(2*kkk-1);
    r1=Rtbt0(2*kkk);
    r2=Rtbt1(2*kkk-1);
    r3=Rtbt1(2*kkk);

    %Combine
    Dtbt(2*kkk-1)=conj(h0(2*kkk))*r0 + h1(2*kkk)*conj(r1) + conj(h2(2*kkk))*r2
    Dtbt(2*kkk)=conj(h1(2*kkk))*r0 - h0(2*kkk)*conj(r1) + conj(h3(2*kkk))*r2 -

end

Ae=Ag.*conj(rchan_flat.PathGains.');
Ae2=Ag2.*conj(rchan_flat2.PathGains.');
Ae3=Ag3.*conj(rchan_flat3.PathGains.');
Ae4=Ag4.*conj(rchan_flat4.PathGains.');
```

```
    %demodulate
    Z=qamdemod(Ae,m);
    Z_mrrc2=qamdemod(Ae+Ae2,m);
    Z_mrrc4=qamdemod(Ae+Ae2+Ae3+Ae4,m);
    Z_new=qamdemod(Dnew,m);
    Z_tbt=qamdemod(Dtbt,m);

    %calculate bit error rate
    BER_reg(k)=biterr(Z,X)/(n);
    BER_mrrc2(k)=biterr(Z_mrrc2,X)/(n);
    BER_mrrc4(k)=biterr(Z_mrrc4,X)/(n);
    BER_new(k)=biterr(Z_new,X)/(n);
    BER_tbt(k)=biterr(Z_tbt,X)/(n);
end

%plots
figure
semilogy(EbNo, BER_reg,'ko-');
hold on;
semilogy(EbNo, BER_mrrc2,'gv-');
semilogy(EbNo, BER_mrrc4,'rs-');
semilogy(EbNo, BER_new,'bd-');
semilogy(EbNo, BER_tbt,'m^-');
xlabel('EbNo (dB)')
ylabel('BER')
title({'The BER performance comparison of coherent BPSK with MRRC'; 'and two-branc
legend('no diversity(1 Tx, 1 Rx)', 'MRRC(1 Tx, 2 Rx)', 'MRRC(1 Tx, 4 Rx)','new sch
```



Published with MATLAB® R2013a