16-5-2022

Wireless Communications

Problem 3

Name: 乌斯曼 Usman

Teacher Name: Du Bing

Student ID: M202161031

计算机与通信工程学院

University of Science and Technology Beijing

16-5-2022

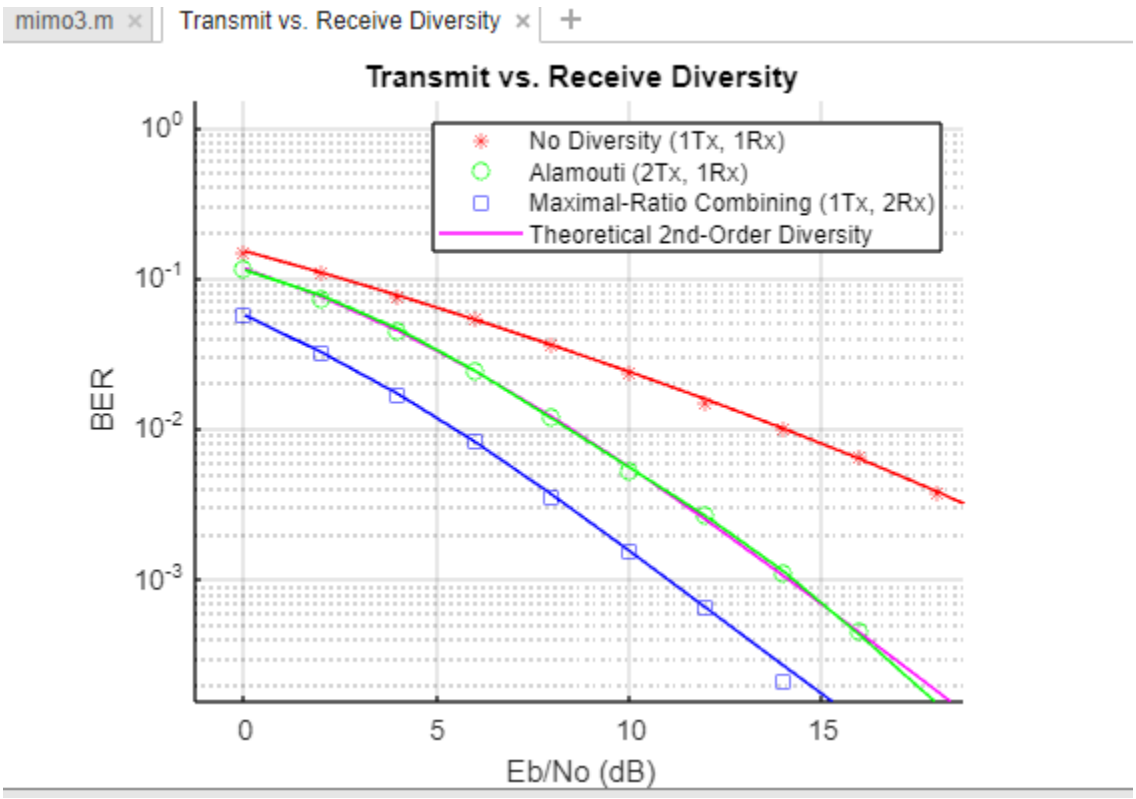
Code 1:

```
frmLen = 100;       % frame length
numPackets = 1000;  % number of packets
EbNo = 0:2:20;      % Eb/No varying to 20 dB
N = 2;              % maximum number of Tx antennas
M = 2;   % Set up a figure for visualizing BER results
% Create comm.BPSKModulator and comm.BPSKDemodulator System objects(TM)
P = 2;                    % modulation order
bpskMod = comm.BPSKModulator;
bpskDemod = comm.BPSKDemodulator('OutputDataType','double');

% Create comm.OSTBCEncoder and comm.OSTBCCombiner System objects
ostbcEnc = comm.OSTBCEncoder;
ostbcComb = comm.OSTBCCombiner;

% Convert Eb/No values to SNR values. The output of the BPSK modulator
% generates unit power signals.
SNR = convertSNR(EbNo,"ebno","BitsPerSymbol",1);

% Create comm.ErrorRate calculator System objects to evaluate BER.
errorCalc1 = comm.ErrorRate;
errorCalc2 = comm.ErrorRate;
errorCalc3 = comm.ErrorRate;

% Since the AWGN function as well as the RANDI function use the default
% random stream, the following commands are executed so that the results
% will be repeatable, i.e., same results will be obtained for every run of
% the example. The default stream will be restored at the end of the
% example.
s = rng(55408);

% Pre-allocate variables for speed
H = zeros(frmLen, N, M);
ber_noDiver  = zeros(3,length(EbNo));
ber_Alamouti = zeros(3,length(EbNo));
ber_MaxRatio = zeros(3,length(EbNo));
ber_thy2     = zeros(1,length(EbNo));
fig = figure;
grid on;
ax = fig.CurrentAxes;
hold(ax,'on');

ax.YScale = 'log';
xlim(ax,[EbNo(1), EbNo(end)]);
ylim(ax,[1e-4 1]);
xlabel(ax,'Eb/No (dB)');
ylabel(ax,'BER');
fig.NumberTitle = 'off';
fig.Renderer = 'zbuffer';
fig.Name = 'Transmit vs. Receive Diversity';
title(ax,'Transmit vs. Receive Diversity');
set(fig, 'DefaultLegendAutoUpdate', 'off');
fig.Position = figposition([15 50 25 30]);
```

```
% Loop over several EbNo points
for idx = 1:length(EbNo)
    reset(errorCalc1);
    reset(errorCalc2);
    reset(errorCalc3);
    % Loop over the number of packets
    for packetIdx = 1:numPackets
        % Generate data vector per frame
        data = randi([0 P-1], frmLen, 1);

        % Modulate data
        modData = bpskMod(data);

        % Alamouti Space-Time Block Encoder
        encData = ostbcEnc(modData);

        % Create the Rayleigh distributed channel response matrix
        %   for two transmit and two receive antennas
        H(1:N:end, :, :) = (randn(frmLen/2, N, M) + ...
                            1i*randn(frmLen/2, N, M))/sqrt(2);
        %   assume held constant for 2 symbol periods
        H(2:N:end, :, :) = H(1:N:end, :, :);

        % Extract part of H to represent the 1x1, 2x1 and 1x2 channels
        H11 = H(:,1,1);
        H21 = H(:,:,1)/sqrt(2);
        H12 = squeeze(H(:,1,:));

        % Pass through the channels
        chanOut11 = H11 .* modData;
        chanOut21 = sum(H21.* encData, 2);
        chanOut12 = H12 .* repmat(modData, 1, 2);

        % Add AWGN
        rxSig11 = awgn(chanOut11,SNR(idx));
        rxSig21 = awgn(chanOut21,SNR(idx));
        rxSig12 = awgn(chanOut12,SNR(idx));

        % Alamouti Space-Time Block Combiner
        decData = ostbcComb(rxSig21, H21);

        % ML Detector (minimum Euclidean distance)
        demod11 = bpskDemod(rxSig11.*conj(H11));
        demod21 = bpskDemod(decData);
        demod12 = bpskDemod(sum(rxSig12.*conj(H12), 2));

        % Calculate and update BER for current EbNo value
        %   for uncoded 1x1 system
        ber_noDiver(:,idx)  = errorCalc1(data, demod11);
        %   for Alamouti coded 2x1 system
        ber_Alamouti(:,idx) = errorCalc2(data, demod21);
        %   for Maximal-ratio combined 1x2 system
        ber_MaxRatio(:,idx) = errorCalc3(data, demod12);

    end % end of FOR loop for numPackets
```

```
    % Calculate theoretical second-order diversity BER for current EbNo
    ber_thy2(idx) = berfading(EbNo(idx), 'psk', 2, 2);

    % Plot results
    semilogy(ax,EbNo(1:idx), ber_noDiver(1,1:idx), 'r*', ...
             EbNo(1:idx), ber_Alamouti(1,1:idx), 'go', ...
             EbNo(1:idx), ber_MaxRatio(1,1:idx), 'bs', ...
             EbNo(1:idx), ber_thy2(1:idx), 'm');
    legend(ax,'No Diversity (1Tx, 1Rx)', 'Alamouti (2Tx, 1Rx)',...
           'Maximal-Ratio Combining (1Tx, 2Rx)', ...
           'Theoretical 2nd-Order Diversity');

    drawnow;
end  % end of for loop for EbNo

% Perform curve fitting and replot the results
fitBER11 = berfit(EbNo, ber_noDiver(1,:));
fitBER21 = berfit(EbNo, ber_Alamouti(1,:));
fitBER12 = berfit(EbNo, ber_MaxRatio(1,:));
semilogy(ax,EbNo, fitBER11, 'r', EbNo, fitBER21, 'g', EbNo, fitBER12, 'b');
hold(ax,'off');

% Restore default stream
rng(s);
```

Output1:



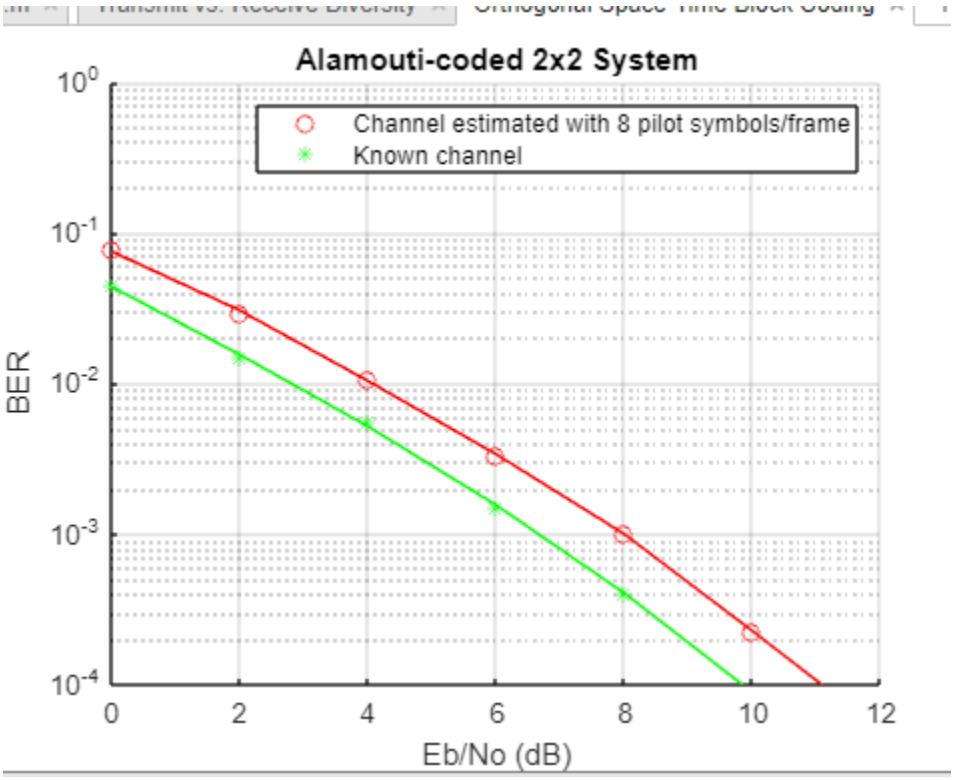
Code 2:

```
frmLen = 100;           % frame length
maxNumErrs = 300;       % maximum number of errors
maxNumPackets = 3000;   % maximum number of packets
EbNo = 0:2:12;          % Eb/No varying to 12 dB
N = 2;                  % number of Tx antennas
M = 2;                  % number of Rx antennas
pLen = 8;               % number of pilot symbols per frame
W = hadamard(pLen);
pilots = W(:, 1:N);     % orthogonal set per transmit antenna
% Create a comm.MIMOChannel System object to simulate the 2x2 spatially
% independent flat-fading Rayleigh channel
chan = comm.MIMOChannel( ...
    'MaximumDopplerShift', 0, ...
    'SpatialCorrelationSpecification', 'None', ...
    'NumTransmitAntennas', N, ...
    'NumReceiveAntennas', M, ...
    'PathGainsOutputPort', true);

% Change the NumReceiveAntennas property value of the hAlamoutiDec System
% object to M that is 2
release(ostbcComb);
ostbcComb.NumReceiveAntennas = M;

% Set the global random stream for repeatability
s = rng(55408);

% Pre-allocate variables for speed
HEst = zeros(frmLen, N, M);
ber_Estimate = zeros(3,length(EbNo));
ber_Known    = zeros(3,length(EbNo));
% Set up a figure for visualizing BER results
fig = figure;
grid on;
ax = fig.CurrentAxes;
hold(ax,'on');

ax.YScale = 'log';
xlim(ax,[EbNo(1), EbNo(end)]);
ylim(ax,[1e-4 1]);
xlabel(ax,'Eb/No (dB)');
ylabel(ax,'BER');
fig.NumberTitle = 'off';
fig.Name = 'Orthogonal Space-Time Block Coding';
fig.Renderer = 'zbuffer';
title(ax,'Alamouti-coded 2x2 System');
set(fig,'DefaultLegendAutoUpdate','off');
fig.Position = figposition([41 50 25 30]);

% Loop over several EbNo points
for idx = 1:length(EbNo)
    reset(errorCalc1);
    reset(errorCalc2);
```

```
    % Loop till the number of errors exceed 'maxNumErrs'
    % or the maximum number of packets have been simulated
    while (ber_Estimate(2,idx) < maxNumErrs) && ...
          (ber_Known(2,idx) < maxNumErrs) && ...
          (ber_Estimate(3,idx)/frmLen < maxNumPackets)
        % Generate data vector per frame
        data = randi([0 P-1], frmLen, 1);

        % Modulate data
        modData = bpskMod(data);

        % Alamouti Space-Time Block Encoder
        encData = ostbcEnc(modData);

        % Prepend pilot symbols for each frame
        txSig = [pilots; encData];

        % Pass through the 2x2 channel
        reset(chan);
        [chanOut, H] = chan(txSig);

        % Add AWGN
        rxSig = awgn(chanOut,SNR(idx));

        % Channel Estimation
        %   For each link => N*M estimates
        HEst(1,:,:) = pilots(:,:).' * rxSig(1:pLen, :) / pLen;
        %   assume held constant for the whole frame
        HEst = HEst(ones(frmLen, 1), :, :);

        % Combiner using estimated channel
        decDataEst = ostbcComb(rxSig(pLen+1:end,:), HEst);

        % Combiner using known channel
        decDataKnown = ostbcComb(rxSig(pLen+1:end,:), ...
                           squeeze(H(pLen+1:end,:,:,:)));

        % ML Detector (minimum Euclidean distance)
        demodEst   = bpskDemod(decDataEst);      % estimated
        demodKnown = bpskDemod(decDataKnown);    % known

        % Calculate and update BER for current EbNo value
        %   for estimated channel
        ber_Estimate(:,idx) = errorCalc1(data, demodEst);
        %   for known channel
        ber_Known(:,idx)    = errorCalc2(data, demodKnown);

    end % end of FOR loop for numPackets

    % Plot results
    semilogy(ax,EbNo(1:idx), ber_Estimate(1,1:idx), 'ro');
    semilogy(ax,EbNo(1:idx), ber_Known(1,1:idx), 'g*');
    legend(ax,['Channel estimated with ' num2str(pLen) ' pilot symbols/frame'],...
           'Known channel');
    drawnow;
```

```
end  % end of for loop for EbNo

% Perform curve fitting and replot the results
fitBEREst   = berfit(EbNo, ber_Estimate(1,:));
fitBERKnown = berfit(EbNo, ber_Known(1,:));
semilogy(ax,EbNo, fitBEREst, 'r', EbNo, fitBERKnown, 'g');
hold(ax,'off');

% Restore default stream
rng(s)
```

Output2:



END