Faculty of Engineering – Cairo University

Electronics and Electrical Communications Engineering Department

Fourth Year – Main Stream

Second Semester - Academic Year 2024/2025

Assignment #1

Space Time Block Codes

Submitted to
Dr. Mai Kafafy

Submitted by

NAME	SEC	ID
مريم احمد كمال ممدوح	4	9211131
مريم حسام عبدالغفار محمد	4	9211138

Table of Contents

Hand Analysis – Decoding the received signal

- **STBC A**

→ We have 3 antennas send over 4 time slots, and each time slot the Rx receives 1 symbol:

Tx Antenna 1	s_1	$-s_2^*$	0	0
Tx Antenna 2	s_2	s_1^*	0	0
Tx Antenna 3	0	0	s_1	$-s_2^*$
	t_1	t_2	t_3	t_4

→ Since there are 2 symbols sent over 4 time slots:

$$Rate = \frac{\#symbols}{\#time\ slots} = \frac{2}{4} = \frac{1}{2}$$

→ Let the channel between each transmitter antenna and the receiver antenna is h_1, h_2, h_3.

→ Now, get the received symbol in terms of transmitted signal and the channel:

$$y_1 = s_1 h_1 + s_2 h_2 \quad \rightarrow y_1 = s_1 h_1 + s_2 h_2$$

$$y_2 = -s_2^* h_1 + s_1^* h_2 \rightarrow y_2^* = s_1 h_2^* - s_2 h_1^*$$

$$y_3 = s_1 h_3 \qquad \rightarrow y_3 = s_1 h_3$$

$$y_4 = -s_2^* h_3 \qquad \rightarrow y_4^* = -s_2 h_3^*$$

→ Write the matrix representation of the equations:

$$\begin{bmatrix} y_1 \\ y_2^* \\ y_3 \\ y_4^* \end{bmatrix} = \begin{bmatrix} h_1 & h_2 \\ h_2^* & -h_1^* \\ h_3 & 0 \\ 0 & -h_3^* \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix}$$

→ Decode the received signal:

$$s_1 : \frac{C_1^{*T}}{\|C_1\|^2} * (C_1 \quad C_2)\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} + \underline{n} * \frac{C_1^{*T}}{\|C_1\|^2} = \frac{\|C_1\|^2}{\|C_1\|^2} * s_1 + 0 + \underline{n} * \frac{C_1^{*T}}{\|C_1\|^2} = \boxed{s_1 + \underline{n}.\frac{C_1^{*T}}{\|C_1\|^2}}$$

$$s_2 : \frac{C_2^{*T}}{\|C_2\|^2} * (C_1 \quad C_2)\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} + \underline{n} * \frac{C_2^{*T}}{\|C_2\|^2} = \frac{\|C_2\|^2}{\|C_2\|^2} * s_2 + 0 + \underline{n} * \frac{C_2^{*T}}{\|C_2\|^2} = \boxed{s_2 + \underline{n}.\frac{C_2^{*T}}{\|C_2\|^2}}$$

Where: C_1& C_2 are the first and second columns of the channel matrix.

- **STBC B**

Tx Antenna 1	s_1	$-s_2^*$	s_3^*	0
Tx Antenna 2	s_2	s_1^*	0	s_3^*
Tx Antenna 3	s_3	0	$-s_1^*$	$-s_2^*$
	t_1	t_2	t_3	t_4

→ Since there are 3 symbols sent over 4 time slots:

$$Rate = \frac{\#symbols}{\#time\ slots} = \frac{3}{4}$$

⇒ Rate in this case is higher, leading to higher BER.

→ Let the channel between each transmitter antenna and the receiver antenna is h_1, h_2, h_3.

→ Now, get the received symbol in terms of transmitted signal and the channel:

$$y_1 = s_1h_1 + s_2h_2 + s_3h_3 \quad \rightarrow y_1 = s_1h_1 + s_2h_2 + s_3h_3$$

$$y_2 = -s_2^*h_1 + s_1^*h_2 \quad \rightarrow y_2^* = s_1h_2^* + s_2(-h_1^*)$$

$$y_3 = s_3^*h_1 - s_1^*h_3 \quad \rightarrow y_3^* = s_1(-h_3^*) + s_3h_1^*$$

$$y_4 = s_3^*h_2 - s_2^*h_3 \quad \rightarrow y_4^* = s_2(-h_3^*) + s_3h_2^*$$

→ Write the matrix representation of the equations:

$$\begin{bmatrix} y_1 \\ y_2^* \\ y_3^* \\ y_4^* \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_2^* & -h_1^* & 0 \\ -h_3^* & 0 & h_1^* \\ 0 & -h_3^* & h_2^* \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix}$$

→ Decode the received signal:

$$s_1: \frac{C_1^{*T}}{\|C_1\|^2} * (C_1 \quad C_2 \quad C_3)\begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} + \underline{n} * \frac{C_1^{*T}}{\|C_1\|^2} = \frac{\|C_1\|^2}{\|C_1\|^2} * s_1 + 0 + \underline{n} * \frac{C_1^{*T}}{\|C_1\|^2} = \boxed{s_1 + \underline{n}.\frac{C_1^{*T}}{\|C_1\|^2}}$$

$$s_2: \frac{C_2^{*T}}{\|C_2\|^2} * (C_1 \quad C_2 \quad C_3)\begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} + \underline{n} * \frac{C_2^{*T}}{\|C_2\|^2} = \frac{\|C_2\|^2}{\|C_2\|^2} * s_2 + 0 + \underline{n} * \frac{C_2^{*T}}{\|C_2\|^2} = \boxed{s_2 + \underline{n}.\frac{C_2^{*T}}{\|C_2\|^2}}$$

$$s_3: \frac{C_3^{*T}}{\|C_3\|^2} * (C_1 \quad C_2 \quad C_3)\begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} + \underline{n} * \frac{C_3^{*T}}{\|C_3\|^2} = \frac{\|C_3\|^2}{\|C_3\|^2} * s_3 + 0 + \underline{n} * \frac{C_3^{*T}}{\|C_3\|^2} = \boxed{s_3 + \underline{n}.\frac{C_3^{*T}}{\|C_3\|^2}}$$

Where: C_1, C_2, and C_3 are the first and second columns of the channel matrix.

Questions Answers

- **Which of the two codes could be considered an Orthogonal STBC? And why?**

  $\rightarrow$ Both of them are orthogonal.

$\Rightarrow$ Check orthogonality of STBC (A):

$$\Rightarrow \begin{bmatrix} h_1 & h_2 \\ h_2^* & -h_1^* \\ h_3 & 0 \\ 0 & -h_3^* \end{bmatrix}$$

$$C_1^{*T} C_2 = (h_1^* \quad h_2 \quad h_3^* \quad 0) * \begin{pmatrix} h_2 \\ -h_1^* \\ 0 \\ -h_3^* \end{pmatrix} = h_1^* h_2 - h_2 h_1^* + 0 + 0 = 0$$

$\therefore$ STBC (A) is orthogonal.

$\Rightarrow$ Check orthogonality of STBC (B):

$$\Rightarrow \begin{bmatrix} h_1 & h_2 & h_3 \\ h_2^* & -h_1^* & 0 \\ -h_3^* & 0 & h_1^* \\ 0 & -h_3^* & h_2^* \end{bmatrix}$$

$$C_1^{*T} C_2 = (h_1^* \quad h_2 \quad -h_3 \quad 0) * \begin{pmatrix} h_2 \\ -h_1^* \\ 0 \\ -h_3^* \end{pmatrix} = h_1^* h_2 - h_2 h_1^* - 0 - 0 = 0$$

$$C_2^{*T} C_3 = (h_2^* \quad -h_1 \quad 0 \quad -h_3) * \begin{pmatrix} h_3 \\ 0 \\ h_1^* \\ h_2^* \end{pmatrix} = h_2^* h_3 - 0 + 0 - h_3 h_2^* = 0$$

$$C_1^{*T} C_3 = (h_1^* \quad h_2 \quad -h_3 \quad 0) * \begin{pmatrix} h_3 \\ 0 \\ h_1^* \\ h_2^* \end{pmatrix} = h_1^* h_3 + 0 - h_3 h_1^* + 0 = 0$$

$\therefore$ STBC (A) is orthogonal.

- **How can you decode the received signal of the two codes.**

  $\rightarrow$ Decoding is done in the hand analysis.

- **Can the receiver decode the received symbols without knowing the channel? Why?**

  $\rightarrow$ Yes, Because the STBC first estimates the channel using pilot symbols sent before the data, then it assumes the channel to be static during the total period used to transmit the symbols by making sure that all 4 time slots are sent within the same coherence time of the channel, hence channel parameters are already assumed to be known to the receiver.

- **Performance Metrics Comparison**

Metric / Code-word	STBC (A)	STBC (B)
Diversity Order	3 (since each symbol hits all 3 antennas across time)	2 (less spatial redundancy due to symbol overlap)
Average symbol rate	$Rate = \frac{1}{2}$ symbols/timeslot (4 time slots for 2 symbols)	$Rate = \frac{3}{4}$ symbols/timeslot (4 time slots for 3 symbols)

Diversity Gain

→ The SNR values are calculated for each code at the same BER, then subtracted from each other to get the diversity gain.

→ Diversity gain = SNR (A) – SNR (B)

```
Diversity Gain at BER = 1.0e-04: 2.03 dB
Diversity Gain at BER = 1.0e-05: 2.54 dB
```

Figure 1 - Diversity gain calculated with MATLAB.

→ The STBC (A) has higher SNR than the STBC (B) at the same BER, so it has higher diversity gain.

→ Diversity gain is chosen to be calculated at relatively higher SNR values (lower BER) where fading is the dominant error cause and the slope directly reflects the diversity order.

- **Which STBC code would you use in a power-limited scenario? And why?**

→ Use STBC (A).

→ Because in power-limited scenarios, we must prioritize reliability over data rate. This means achieving low BER with low SNR to stay within the power budget. Even though this will cause BW inefficiency due to low rate of the system.

→ STBC (A) gives full diversity and simpler decoding, which improves error performance at low SNR.

Results of system simulation

- ***BER Comparison***

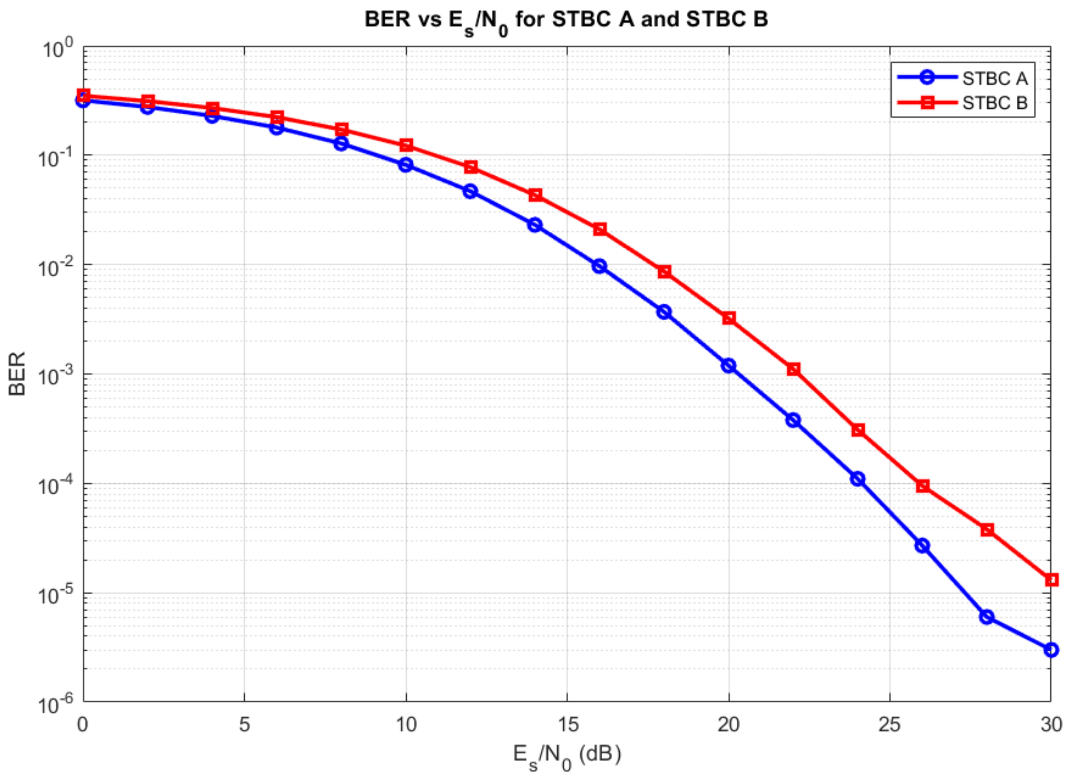

Figure 2 - BER Comparison between STBC (A) & STBC (B).

Comment

→ BER of STBC A is lower (better) than BER of STBC B.

→ We use the same total power for both codes, and since STBC (A) transmits only 2 symbols over 4 timeslots while STBC (B) transmits 3 symbols, this means STBC (A) has higher energy per symbol which makes the symbol more robust to noise and fading, and thus achieving lower BER.

→ Also, STBC (A) has better diversity utilization, because each symbol is sent over all available channel paths at each timeslot.

- ***Diversity order verification for code A***

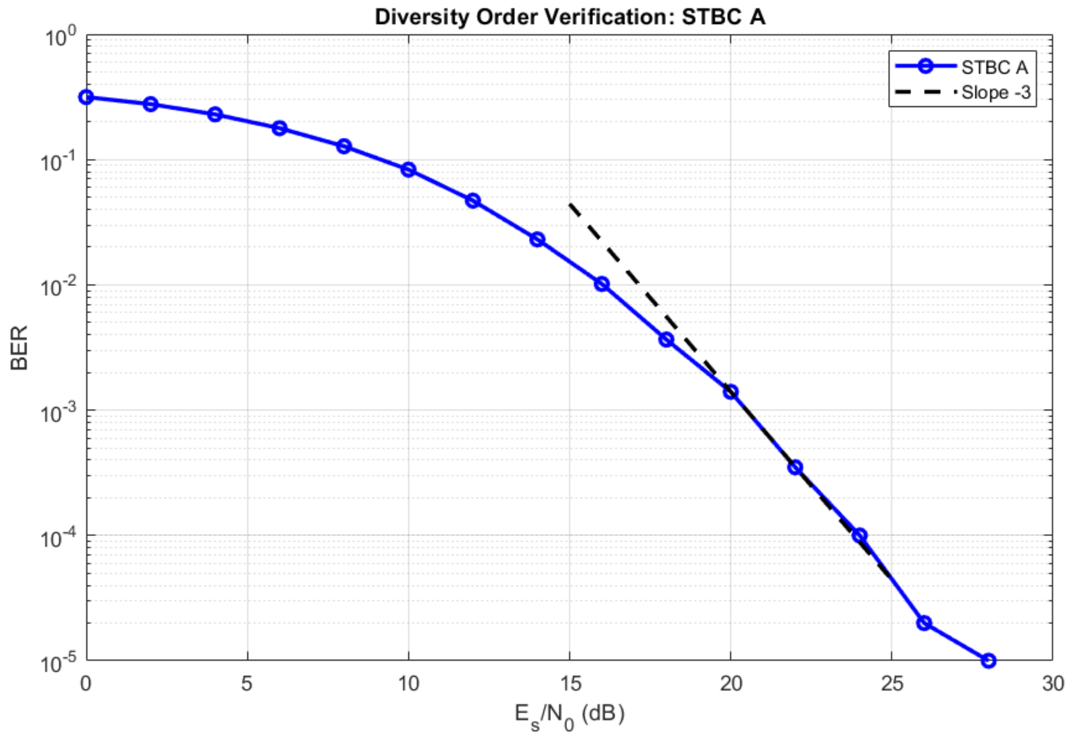

Figure 3 - Diversity order verification for STBC (A) when number of bits is 1e5.

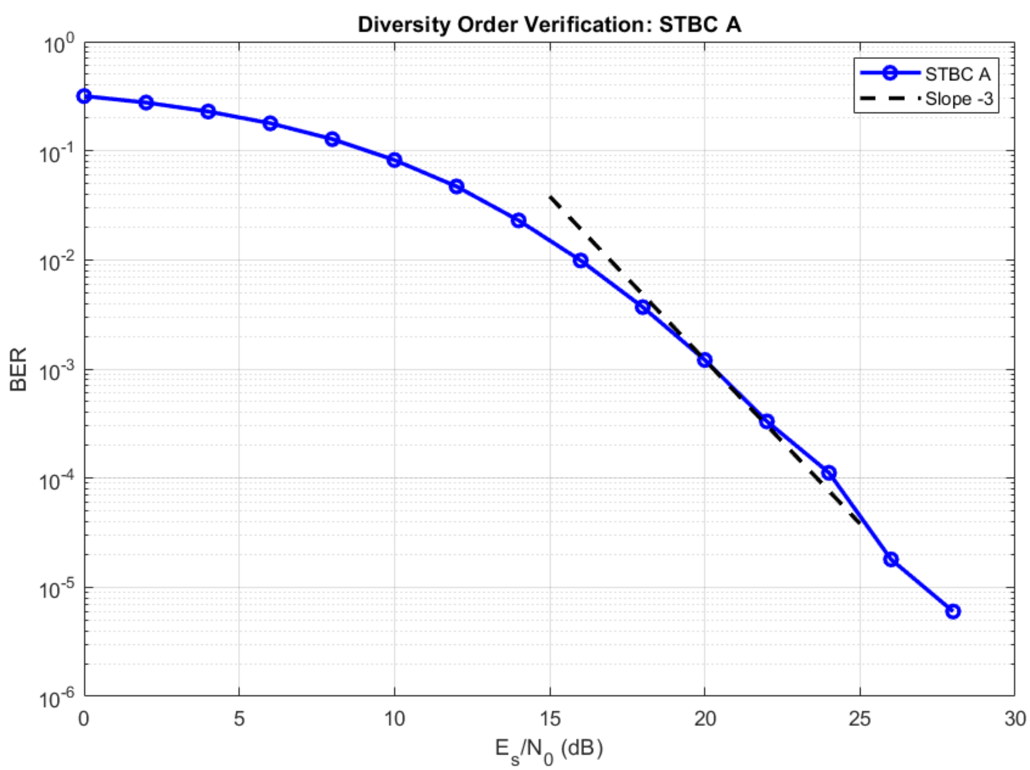

Figure 4 - Diversity order verification for STBC (B) with number of bits (1e6).

Observation:

The slope of the BER curve of STBC A approximately matches the -3 slope at higher SNR, validating the diversity order = 3.

- ***Diversity order verification for code B***

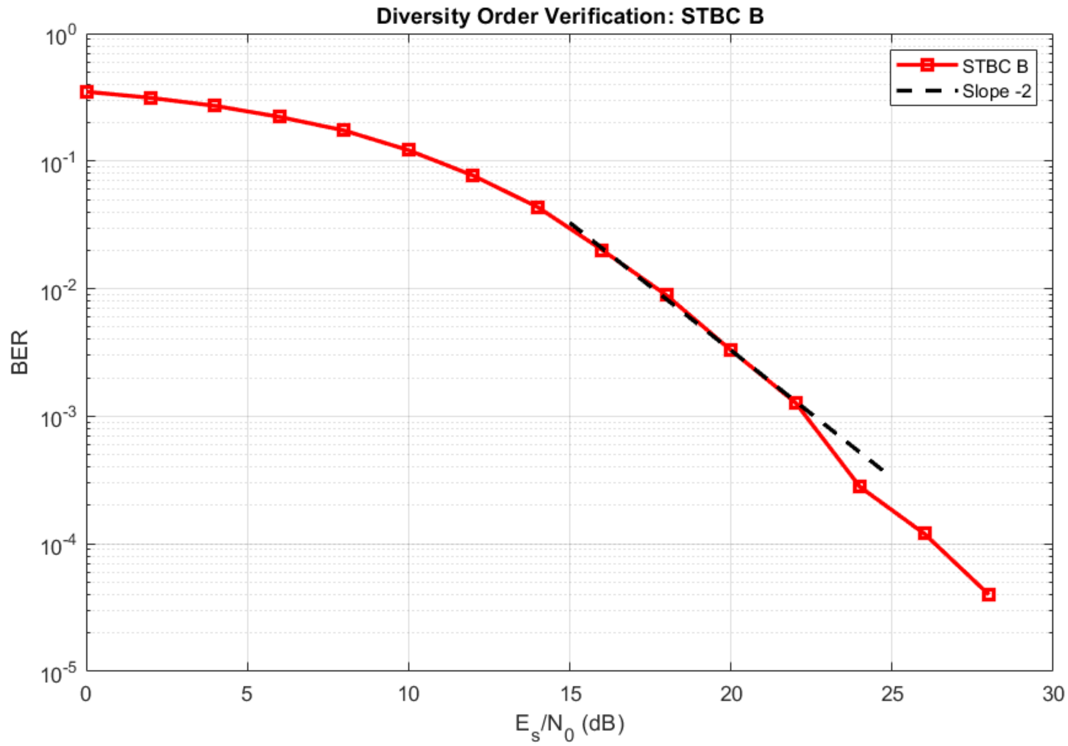

Figure 5 - Diversity order verification for STBC (B) when number of bits is 1e5.

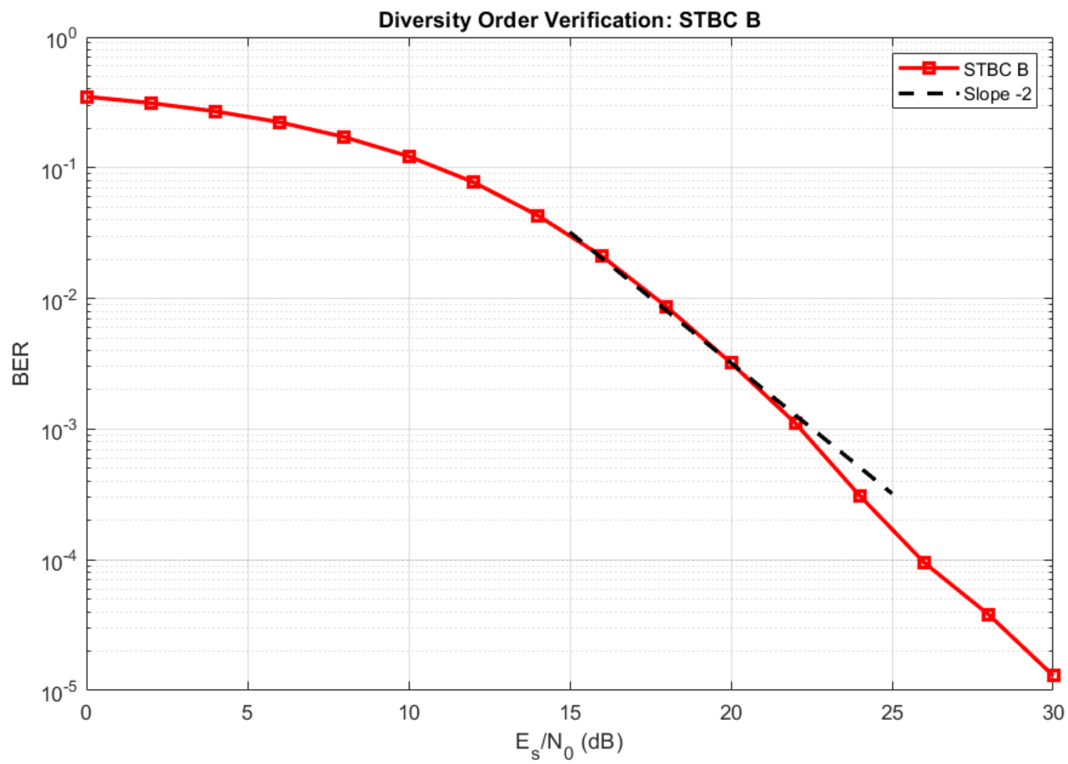

Figure 6 - Diversity order verification for STBC (B) with higher number of bits (1e6).

<u>Observation</u>:

The slope of the BER curve of STBC B approximately matches the -2 slope, validating the diversity order = 2.

The MATLAB Code

```
clear all; close all; clc;
%% Simulation Parameters
N = 1e6;                       % Number of bits
M = 4;                         % QPSK modulation
k = log2(M);                   % Bits per symbol
A = 1;                         % QPSK Amplitude
Eb = A^2;
Es = 2 * Eb;                   % QPSK: 2 bits/symbol
numSymbols = N/k;
EsN0_dB = 0:2:30;              % Es/N0 in dB
EsN0_lin = 10.^(EsN0_dB/10); % Es/N0 linear
numAntennas = 3;               % Transmit antennas
numRxAntennas = 1;             % Receive antennas

%% QPSK Modulation
qpskMod = comm.QPSKModulator('BitInput',true);
qpskDemod = comm.QPSKDemodulator('BitOutput',true);

%% STBC Encoding Functions
encodeSTBC_A = @(s) [
    s(1)      s(2)      0;
   -conj(s(2)) conj(s(1))  0;
    0         0         s(1);
    0         0        -conj(s(2))
];

encodeSTBC_B = @(s) [
    s(1)      s(2)      s(3);
   -conj(s(2)) conj(s(1))  0;
    conj(s(3)) 0      -conj(s(1));
    0         conj(s(3)) -conj(s(2))
];

%% BER Initialization
ber_STBC_A = zeros(size(EsN0_dB));
ber_STBC_B = zeros(size(EsN0_dB));

%% Simulation Loop
for idx = 1:length(EsN0_dB)
    errors_STBC_A = 0;
    errors_STBC_B = 0;
    totalBits = 0;

    while totalBits < N
        %% Generate random bits and QPSK modulation
        txBits = randi([0 1], N, 1);
        txSym = qpskMod(txBits);

        %% Split into blocks
        numBlocks_A = floor(length(txSym)/2);
        numBlocks_B = floor(length(txSym)/3);

        %% STBC A Encoding
        txSym_A = txSym(1:2*numBlocks_A);
```

```
        txSym_A = reshape(txSym_A, 2, numBlocks_A);
        encoded_A = zeros(4, numAntennas, numBlocks_A);
        for n = 1:numBlocks_A
            block = encodeSTBC_A(txSym_A(:,n));
            block = block / sqrt(norm(block,'fro')^2);
            encoded_A(:,:,n) = block;
        end

        %% STBC B Encoding
        txSym_B = txSym(1:3*numBlocks_B);
        txSym_B = reshape(txSym_B, 3, numBlocks_B);
        encoded_B = zeros(4, numAntennas, numBlocks_B);
        for n = 1:numBlocks_B
            block = encodeSTBC_B(txSym_B(:,n));
            block = block / sqrt(norm(block,'fro')^2);
            encoded_B(:,:,n) = block;
        end

        %% Improved Block Rayleigh Flat Fading Channel
        H_A = (randn(numAntennas, numBlocks_A) + 1i*randn(numAntennas,
numBlocks_A)) / sqrt(2);
        H_B = (randn(numAntennas, numBlocks_B) + 1i*randn(numAntennas,
numBlocks_B)) / sqrt(2);

        %% AWGN Noise
        N0 = Es ./ EsN0_lin(idx);

        %% Transmission - STBC A
        rx_A = zeros(4, 1, numBlocks_A);
        for n = 1:numBlocks_A
            X = squeeze(encoded_A(:,:,n));      % 4x3
            h = H_A(:,n);                       % 3x1
            noise = sqrt(N0/2) * (randn(4,1) + 1i*randn(4,1));
            rx_A(:,:,n) = X * h + noise;        % 4x1
        end

        %% Transmission - STBC B
        rx_B = zeros(4, 1, numBlocks_B);
        for n = 1:numBlocks_B
            X = squeeze(encoded_B(:,:,n));      % 4x3
            h = H_B(:,n);                       % 3x1
            noise = sqrt(N0/2) * (randn(4,1) + 1i*randn(4,1));
            rx_B(:,:,n) = X * h + noise;        % 4x1
        end

        %% STBC A Decoding
        decSym_A = zeros(2, numBlocks_A);
        for n = 1:numBlocks_A
            h = H_A(:,n);
            y = rx_A(:,:,n);
            y_vec = [y(1); conj(y(2)); y(3); conj(y(4))];

            C1 = [h(1); conj(h(2)); h(3); 0];
            C2 = [h(2); -conj(h(1)); 0; -conj(h(3))];

            s1_hat = (C1' * y_vec) / (norm(C1)^2);
            s2_hat = (C2' * y_vec) / (norm(C2)^2);
```

```
            decSym_A(:,n) = [s1_hat; s2_hat];
        end
        decSym_A = decSym_A(:);

        %% STBC B Decoding
        decSym_B = zeros(3, numBlocks_B);
        for n = 1:numBlocks_B
            h = H_B(:,n);
            y = rx_B(:,:,n);
            y_vec = [y(1); conj(y(2)); conj(y(3)); conj(y(4))];

            C1 = [h(1); conj(h(2)); -conj(h(3)); 0];
            C2 = [h(2); -conj(h(1)); 0; -conj(h(3))];
            C3 = [h(3); 0; conj(h(1)); conj(h(2))];

            s1_hat = (C1'*y_vec) / (norm(C1)^2);
            s2_hat = (C2'*y_vec) / (norm(C2)^2);
            s3_hat = (C3'*y_vec) / (norm(C3)^2);

            decSym_B(:,n) = [s1_hat; s2_hat; s3_hat];
        end
        decSym_B = decSym_B(:);

        %% QPSK Demodulation
        rxBits_A = qpskDemod(decSym_A);
        rxBits_B = qpskDemod(decSym_B);

        %% BER Calculation
        numBits_A = min(length(txBits), length(rxBits_A));
        numBits_B = min(length(txBits), length(rxBits_B));

        errors_STBC_A = errors_STBC_A + sum(txBits(1:numBits_A) ~=
rxBits_A(1:numBits_A));
        errors_STBC_B = errors_STBC_B + sum(txBits(1:numBits_B) ~=
rxBits_B(1:numBits_B));

        totalBits = totalBits + numBits_A;
    end

    ber_STBC_A(idx) = errors_STBC_A / N;
    ber_STBC_B(idx) = errors_STBC_B / N;

    fprintf('Es/N0 = %2d dB: STBC A BER = %.4e | STBC B BER = %.4e\n', ...
            EsN0_dB(idx), ber_STBC_A(idx), ber_STBC_B(idx));
end

%% Diversity Gain
% Target BER points
targetBERs = [1e-4, 1e-5];

% Loop over targets
for i = 1:length(targetBERs)
    targetBER = targetBERs(i);

    % Find valid index ranges for interpolation (BER > 0 and BER > target)
    validIdx_A = (ber_STBC_A > 0) & (ber_STBC_A >= targetBER);
    validIdx_B = (ber_STBC_B > 0) & (ber_STBC_B >= targetBER);
```

```
    if sum(validIdx_A) < 2 || sum(validIdx_B) < 2
        fprintf('Not enough data to interpolate at BER = %.1e\n', targetBER);
        continue;
    end

    % Sort valid points in increasing BER order (x-axis for interp1 must be
monotonic)
    [berA_sorted, idxA] = sort(ber_STBC_A(validIdx_A), 'descend');
    snrA_sorted = EsN0_dB(validIdx_A);
    snrA_sorted = snrA_sorted(idxA);

    [berB_sorted, idxB] = sort(ber_STBC_B(validIdx_B), 'descend');
    snrB_sorted = EsN0_dB(validIdx_B);
    snrB_sorted = snrB_sorted(idxB);

    % Interpolation
    try
        SNR_A = interp1(berA_sorted, snrA_sorted, targetBER, 'linear',
'extrap');
        SNR_B = interp1(berB_sorted, snrB_sorted, targetBER, 'linear',
'extrap');
        diversityGain_dB = SNR_B - SNR_A;
        fprintf('Diversity Gain at BER = %.1e: %.2f dB\n', targetBER,
diversityGain_dB);
    catch ME
        fprintf('Interpolation failed at BER = %.1e: %s\n', targetBER,
ME.message);
    end
end

%% Plot
figure;
semilogy(EsN0_dB, ber_STBC_A, 'b-o', 'LineWidth', 2); hold on;
semilogy(EsN0_dB, ber_STBC_B, 'r-s', 'LineWidth', 2);
xlabel('E_s/N_0 (dB)'); ylabel('BER'); grid on;
title('BER vs E_s/N_0 for STBC A and STBC B');
legend('STBC A', 'STBC B');

%% Plot: Slope for Diversity Order Verification - STBC A
figure;
semilogy(EsN0_dB, ber_STBC_A, 'b-o', 'LineWidth', 2); hold on;
snr_ref_A = 20;
ber_ref_A = interp1(EsN0_dB, ber_STBC_A, snr_ref_A);
ref_snr_A = [snr_ref_A-5, snr_ref_A+5];
ref_ber_A = ber_ref_A * 10.^(-(ref_snr_A - snr_ref_A) * 3 / 10);
semilogy(ref_snr_A, ref_ber_A, '--k', 'LineWidth', 2);
xlabel('E_s/N_0 (dB)');
ylabel('BER');
legend('STBC A', 'Slope -3');
title('Diversity Order Verification: STBC A');
grid on;

%% Plot: Slope for Diversity Order Verification - STBC B
figure;
semilogy(EsN0_dB, ber_STBC_B, 'r-s', 'LineWidth', 2); hold on;
snr_ref_B = 20;
```

```
ber_ref_B = interp1(EsN0_dB, ber_STBC_B, snr_ref_B);
ref_snr_B = [snr_ref_B-5, snr_ref_B+5];
ref_ber_B = ber_ref_B * 10.^(-(ref_snr_B - snr_ref_B) * 2 / 10);
semilogy(ref_snr_B, ref_ber_B, '--k', 'LineWidth', 2);
xlabel('E_s/N_0 (dB)');
ylabel('BER');
legend('STBC B', 'Slope -2');
title('Diversity Order Verification: STBC B');
grid on;

%% Combined Plot: Slope Lines for Diversity Order Verification
figure;

% STBC A data and slope line
semilogy(EsN0_dB, ber_STBC_A, 'b-o', 'LineWidth', 2, 'DisplayName', 'STBC
A');
hold on;

% Reference point for STBC A
snr_ref_A = 20;
ber_ref_A = interp1(EsN0_dB, ber_STBC_A, snr_ref_A);
ref_snr_A = [snr_ref_A-5, snr_ref_A+5];
ref_ber_A = ber_ref_A * 10.^(-(ref_snr_A - snr_ref_A) * 3 / 10);
semilogy(ref_snr_A, ref_ber_A, '--b', 'LineWidth', 2, 'DisplayName', 'Slope -
3 (STBC A)');

% STBC B data and slope line
semilogy(EsN0_dB, ber_STBC_B, 'r-s', 'LineWidth', 2, 'DisplayName', 'STBC
B');

% Reference point for STBC B
snr_ref_B = 20;
ber_ref_B = interp1(EsN0_dB, ber_STBC_B, snr_ref_B);
ref_snr_B = [snr_ref_B-5, snr_ref_B+5];
ref_ber_B = ber_ref_B * 10.^(-(ref_snr_B - snr_ref_B) * 2 / 10);
semilogy(ref_snr_B, ref_ber_B, '--r', 'LineWidth', 2, 'DisplayName', 'Slope -
2 (STBC B)');

% Plot formatting
xlabel('E_s/N_0 (dB)', 'FontSize', 12);
ylabel('BER', 'FontSize', 12);
title('Diversity Order Verification: STBC A vs STBC B', 'FontSize', 12);
legend('Location', 'southwest', 'FontSize', 10);
grid on;
set(gca, 'FontSize', 10);

% Annotation to highlight diversity orders
text(18, 1e-3, 'Diversity Order = 3', 'Color', 'b', 'FontSize', 10);
text(18, 3e-2, 'Diversity Order = 2', 'Color', 'r', 'FontSize', 10);
```