EC6100 – WIRELESS AND MOBILE COMMUNICATION

ASSIGNMENT – 2

IMPLEMENTATION AND ANALYSIS OF FEEDBACK DIVERSITY USING MATLAB SIMULINK

GROUP MEMBERS:

1. 2020/E/034 – DILUSHANTH P.
2. 2020/E/050 – HEMAKANTH N.
3. 2020/E/063 – JEYASINGAM K.
4. 2020/E/197 – MITHULAVAN V.

SEMESTER VI

22 JUL 2024

CONTENTS

1. INTRODUCTION

Wireless communication systems face numerous challenges, including multipath fading, noise, and interference which can degrade signal quality and reliability. This feedback diversity is one of the technique to address this issues.

This report presents the implementation and analysis of a feedback diversity using MATLAB Simulink.

2. THEORETICAL BACKGROUND

2.1 FUNDAMENTALS

Wireless communications systems transmit information through the space using electromagnetic waves. This system includes basic components to transmit information. Such as,

1. Transmitter – generates and modulates the signal
2. Channel – the medium through which the signal propagates
3. Receiver – detects and demodulates the received signal

During the transmission process, signals are subjected to various impairments that can affect their quality and reliability.

2.2 CHANNEL IMPAIRMENTS

1. Multipath fading

Multipath fading occurs when a signal takes multiple paths from the transmitter to the receiver due to reflection, refraction, and scattering. It will result in multiple copies of the signal arriving at the receiver with different delays, amplitudes, and phases. The Rayleigh fading model is used to describe multipath fading in environments where there is no line-of-sight path between the transmitter and receiver.

2.Additive White Gaussian Noise (AWGN)

It's a noise model. It adds white noise with a constant spectral density and a Gaussian distribution of amplitude to the signal passing through the signal.

2.3 DIVERSITY TECHNIQUES

Diversity techniques aim to improve signal quality and reliability of a system by using two or more communication channels with different characteristics.

Common diversity techniques

1. Spatial diversity – using multiple antennas
2. Time diversity – transmitting same information at different time intervals
3. Frequency diversity – transmitting same information on different frequency bands.
4. Polarization diversity – using different polarizations of electromagnetic waves.

2.4 FEEDBACK DIVERSITY

Feedback diversity is a one of the reception methods of spatial diversity. It's an adaptive technique where the information about the channel conditions back to the transmitter. The transmitter then uses this feedback to adjust its transmission parameters.

Advantages:

- Improved signal quality
- Enhanced spectral efficiency
- Reduced power consumption
- Increased system capacity
- Simple to implement

2.5 SISO SYSTEMS

SISO systems use one transmit antenna and one receive antenna. While simpler to implement, SISO systems are more susceptible to fading and interference compared to multiple antenna systems.

2.6 QUANTIZATION

It involves converting a continuous amplitude signal to a discrete amplitude signal. This process is essential for digital communication but here it introduces quantization noise.

2.7 MOVING AVERAGE FILTER

It's a simple low pass filter used for smoothing time series data. It operates by creating a series of averages of different subsets of the full data set, helping to reduce the impact of random variations in the signal.

3.DESIGN AND IMPLEMENTATION

IMPLEMENTATION USING MATLAB SIMULINK

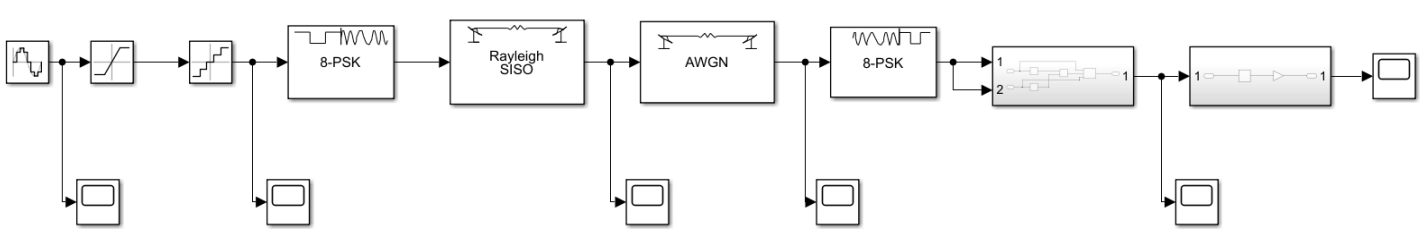

Figure 01: Simulink model for Feedback Diversity

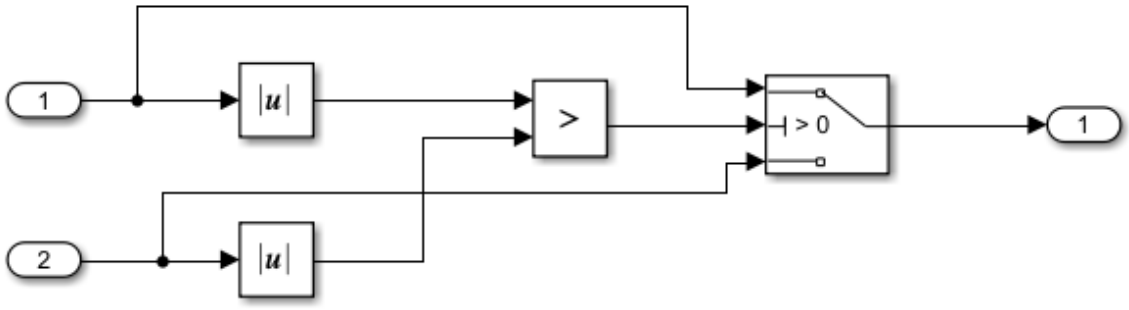

Figure 02: Subsystem for feedback diversity logic

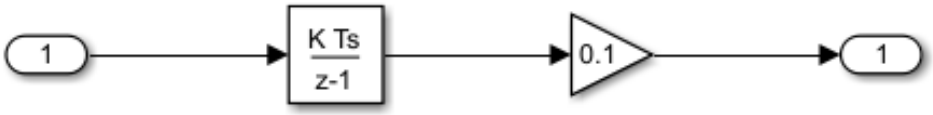

Figure 03: Subsystem for moving average filter

The model consists of the following main components.

1. **Signal generation**
   A sine wave generator was used as the input signal source. This simple waveform allows for easy visualization and analysis of the system's effects on the signal.

2. **Quantization**
   The continuous sine wave was passed through a quantizer to convert it into discrete levels. This step simulates analog to digital conversion process in digital communication systems.

3. **Channel modeling**
   Two channel models were implemented to simulate here.
   I. Rayleigh SISO Channel
      This block simulates the effects of multipath fading in a SISO system. It applies Rayleigh fading to the signal, causing variations in amplitude and phase over time.
   II. AWGN Channel
      This block adds white Gaussian noise to the signal, simulating random noise present in wireless channels.
4. **Feedback Diversity Logic**

   A custom Simulink block was created to implement the feedback diversity logic. This block analysis the channel conditions based on the received signal generates appropriate feedback to the transmitter.

5. **Moving Average Filter**
   A moving average filter was applied to the output signal to smooth out rapid fluctuations of the overall signal.

IMPLEMENTATION USING MATLAB CODE

```
% Define the operating frequency (800 MHz)
freq = 800e6;

% Define the speed of light in m/s
c = physconst("lightspeed");

% Calculate the wavelength in meters
lambda = c / freq;

% Define the bandwidth fraction (10%)
BW_frac = 0.1;

% Calculate the minimum and maximum frequencies
fmin = freq - BW_frac * freq;
fmax = freq + BW_frac * freq;

% Create two identical dipole antennas with half-wavelength length and thin width
d1 = dipole('Length', lambda / 2, 'Width', lambda / 200);
d2 = dipole('Length', lambda / 2, 'Width', lambda / 200);

% Set the spacing between the antennas to 5 wavelengths
range = 5 * lambda;

% Create a linear array with the two dipole elements
l = linearArray;
l.Element = [d1 d2];
l.ElementSpacing = range;

% Visualize the array
show(l);
view(-80, 4);

% Define the number of orientation steps
numpos = 101;

% Check if numpos is a positive integer
if numpos <= 0 || rem(numpos, 1) ~= 0
error('numpos must be a positive integer.');
end

% Define the range of tilt angles from 0 to 90 degrees
orientation = linspace(0, 90, numpos);

% Initialize arrays to store S21 and correlation values
S21_TiltdB = nan(1, numel(orientation));
Corr_TiltdB = nan(1, numel(orientation));

% Set the initial tilt angle
current_tilt = 0;
feedback_step = 1; % Define the feedback step

% Create a figure for feedback iteration analysis
fig1 = figure;

% Loop through each orientation angle
for i = 1:numel(orientation)
% Set the tilt of the second dipole
d2.Tilt = current_tilt;
l.Element(2) = d2;

% Calculate the scattering parameters and correlation
S = sparameters(l, freq);
Corr = correlation(l, freq, 1, 2);

% Store the S21 and correlation values in dB
S21_TiltdB(i) = 20 * log10(abs(S.Parameters(2, 1, 1)));
Corr_TiltdB(i) = 20 * log10(Corr);

% Adjust tilt based on feedback
current_tilt = adjust_tilt_based_on_feedback(current_tilt, S21_TiltdB(i), feedback_step);

% Plot the results
figure(fig1);
plot(orientation, S21_TiltdB, orientation, Corr_TiltdB, 'LineWidth', 2);
grid on;
axis([min(orientation) max(orientation) -65 -20]);
xlabel('Tilt Variation on 2nd Dipole (deg.)');
ylabel('Magnitude (dB)');
title('Correlation, S_{21} Variation with Polarization');
drawnow;
end
```

```
% Add a legend to the plot
legend('S_{21}', 'Correlation');

% Define the number of spacing steps and the range of distances
Nrange = 201;
Rmin = 0.001 * lambda;
Rmax = 2.5 * lambda;
range = linspace(Rmin, Rmax, Nrange);

% Initialize arrays to store S21 and correlation values
S21_RangedB = nan(1, Nrange);
Corr_RangedB = nan(1, Nrange);

% Create a figure for range analysis
fig2 = figure;

% Loop through each spacing distance
for i = 1:Nrange
    % Set the spacing between the elements
    l.ElementSpacing = range(i);

    % Calculate the scattering parameters and correlation
    S = sparameters(l, freq);
    Corr = correlation(l, freq, 1, 2);

    % Store the S21 and correlation values in dB
    S21_RangedB(i) = 20 * log10(abs(S.Parameters(2, 1, 1)));
    Corr_RangedB(i) = 20 * log10(Corr);

    % Plot the results
    figure(fig2);
    plot(range ./ lambda, S21_RangedB, range ./ lambda, Corr_RangedB, '--', 'LineWidth', 2);
    grid on;
    axis([min(range ./ lambda) max(range ./ lambda) -50 0]);
    xlabel('Distance of Separation, d/\lambda');
    ylabel('Magnitude (dB)');
    title('Correlation, S_{21} Variation with Range');
    drawnow;
end

% Add a legend to the plot
legend('S_{21}', 'Correlation');

% Pick a specific separation distance (e.g., 1.25 wavelengths)
Rpick = 1.25 * lambda;

% Define the frequency range for analysis
Numfreq = 101;
f = linspace(fmin, fmax, Numfreq);
l.ElementSpacing = Rpick;

% Calculate the correlation values over the frequency range
Corr_PickdB = nan(1, Numfreq);
for i = 1:Numfreq
    S = sparameters(l, f(i));
    Corr_PickdB(i) = 20 * log10(correlation(l, f(i), 1, 2));
end

% Plot the correlation variation with frequency
fig3 = figure;
plot(f ./ 1e9, Corr_PickdB, 'LineWidth', 2);
grid on;
axis([min(f ./ 1e9) max(f ./ 1e9) -65 0]);
xlabel('Frequency (GHz)');
ylabel('Magnitude (dB)');
title('Correlation Variation with Frequency');

% Additional analyses

% Feedback Iteration Analysis
Tilt_Angles = nan(1, numel(orientation));
current_tilt = 0;
fig4 = figure;
for i = 1:numel(orientation)
    d2.Tilt = current_tilt;
    l.Element(2) = d2;

    S = sparameters(l, freq);
    Corr = correlation(l, freq, 1, 2);

    Tilt_Angles(i) = current_tilt;
    current_tilt = adjust_tilt_based_on_feedback(current_tilt, S21_TiltdB(i), feedback_step); % Default feedback step

    plot(orientation, Tilt_Angles, 'LineWidth', 2);
    grid on;
```

```
    xlabel('Orientation Angle (deg.)');
    ylabel('Tilt Angle (deg.)');
    title('Tilt Angle Adjustment Over Feedback Iterations');
    drawnow;
end

% Feedback Sensitivity Analysis
feedback_steps = [0.5, 1, 2];
fig5 = figure;
for j = 1:numel(feedback_steps)
    S21_FeedbackdB = nan(1, numel(orientation));
    Corr_FeedbackdB = nan(1, numel(orientation));
    current_tilt = 0;

    for i = 1:numel(orientation)
        d2.Tilt = current_tilt;
        l.Element(2) = d2;

        S = sparameters(l, freq);
        Corr = correlation(l, freq, 1, 2);

        S21_FeedbackdB(i) = 20 * log10(abs(S.Parameters(2, 1, 1)));
        Corr_FeedbackdB(i) = 20 * log10(Corr);

        current_tilt = adjust_tilt_based_on_feedback(current_tilt, S21_FeedbackdB(i), feedback_steps(j));
    end

    plot(orientation, S21_FeedbackdB, 'LineWidth', 2);
    hold on;
    plot(orientation, Corr_FeedbackdB, '--', 'LineWidth', 2);
end
grid on;
xlabel('Tilt Variation on 2nd Dipole (deg.)');
ylabel('Magnitude (dB)');
title('S_{21} and Correlation with Different Feedback Adjustment Steps');
legend(arrayfun(@(x) sprintf('Feedback Step = %.1f', x), feedback_steps, 'UniformOutput', false));

% Comparison of Feedback Strategies
feedback_strategies = {@strategy1, @strategy2}; % Define different strategies
fig6 = figure;
for k = 1:numel(feedback_strategies)
    S21_Strategy = nan(1, numel(orientation));
    Corr_Strategy = nan(1, numel(orientation));
    current_tilt = 0;

    for i = 1:numel(orientation)
        d2.Tilt = current_tilt;
        l.Element(2) = d2;

        S = sparameters(l, freq);
        Corr = correlation(l, freq, 1, 2);

        S21_Strategy(i) = 20 * log10(abs(S.Parameters(2, 1, 1)));
        Corr_Strategy(i) = 20 * log10(Corr);

        current_tilt = feedback_strategies{k}(current_tilt, S21_Strategy(i));
    end

    plot(orientation, S21_Strategy, 'LineWidth', 2);
    hold on;
    plot(orientation, Corr_Strategy, '--', 'LineWidth', 2);
end
grid on;
xlabel('Tilt Variation on 2nd Dipole (deg.)');
ylabel('Magnitude (dB)');
title('Performance Metrics for Different Feedback Strategies');
legend(arrayfun(@(x) sprintf('Strategy %d', x), 1:numel(feedback_strategies), 'UniformOutput', false));

% Correlation vs. Range for Various Tilt Angles
tilt_angles = [0, 10, 20, 30];
fig7 = figure;
for m = 1:numel(tilt_angles)
    d2.Tilt = tilt_angles(m);
    l.Element(2) = d2;

    Corr_Range = nan(1, Nrange);

    for i = 1:Nrange
        l.ElementSpacing = range(i);
        Corr = correlation(l, freq, 1, 2);
        Corr_Range(i) = 20 * log10(Corr);
    end

    plot(range ./ lambda, Corr_Range, 'LineWidth', 2);
    hold on;
```

```
end
grid on;
xlabel('Distance of Separation, d/\lambda');
ylabel('Correlation (dB)');
title('Correlation Variation with Separation Distance for Various Tilt Angles');
legend(arrayfun(@(x) sprintf('Tilt Angle = %d', x), tilt_angles, 'UniformOutput', false));

% Function Definitions
function tilt = adjust_tilt_based_on_feedback(current_tilt, S21, feedback_step)
    % Simple feedback adjustment example (you can replace this with your own logic)
    % If the S21 value indicates poor signal strength, increase the tilt angle
    if S21 < -30
        tilt = current_tilt + feedback_step; % Increase tilt angle
    else
        tilt = current_tilt; % Maintain current tilt angle
    end
end

function tilt = strategy1(current_tilt, S21)
    % Strategy 1: Increase tilt if S21 is below -30 dB
    if S21 < -30
        tilt = current_tilt + 1; % Increase tilt by 1 degree
    else
        tilt = current_tilt;
    end
end

function tilt = strategy2(current_tilt, S21)
    % Strategy 2: Decrease tilt if S21 is above -20 dB
    if S21 > -20
        tilt = current_tilt - 1; % Decrease tilt by 1 degree
    else
        tilt = current_tilt;
    end
end
```

4.RESULTS AND ANALYSIS

RESULTS FROM MATLAB SIMULINK

The system's performance was analyzed using multiple scope outputs at different stages.

1. Sine wave output

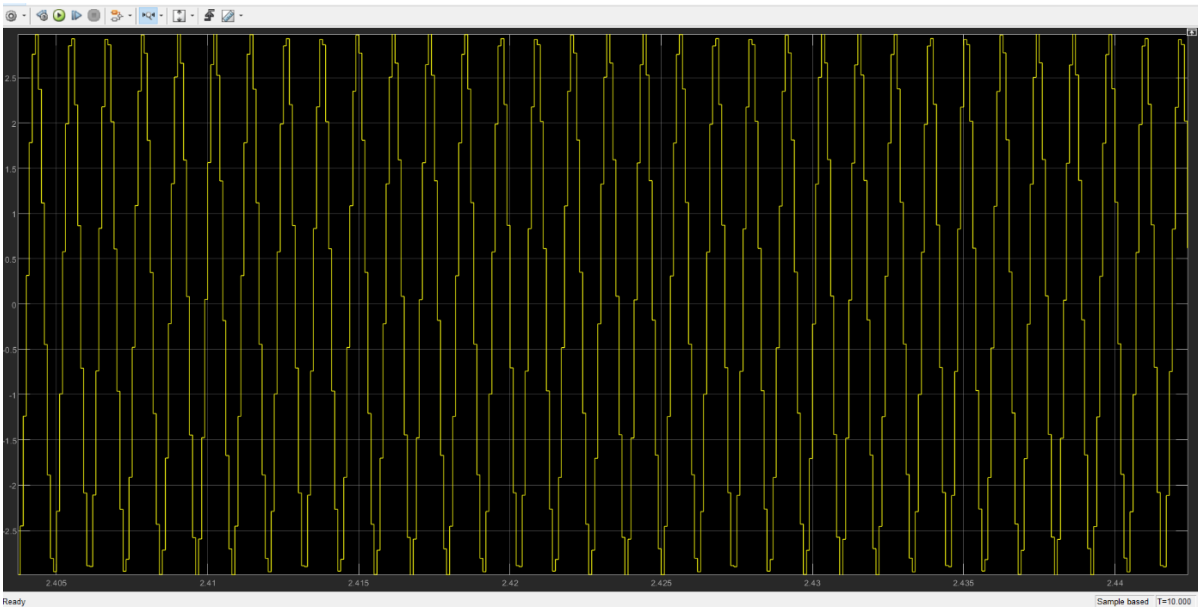

Figure 04: Output from sine wave input signal

- Output shows a input signal of sine wave

2. Quantizer output

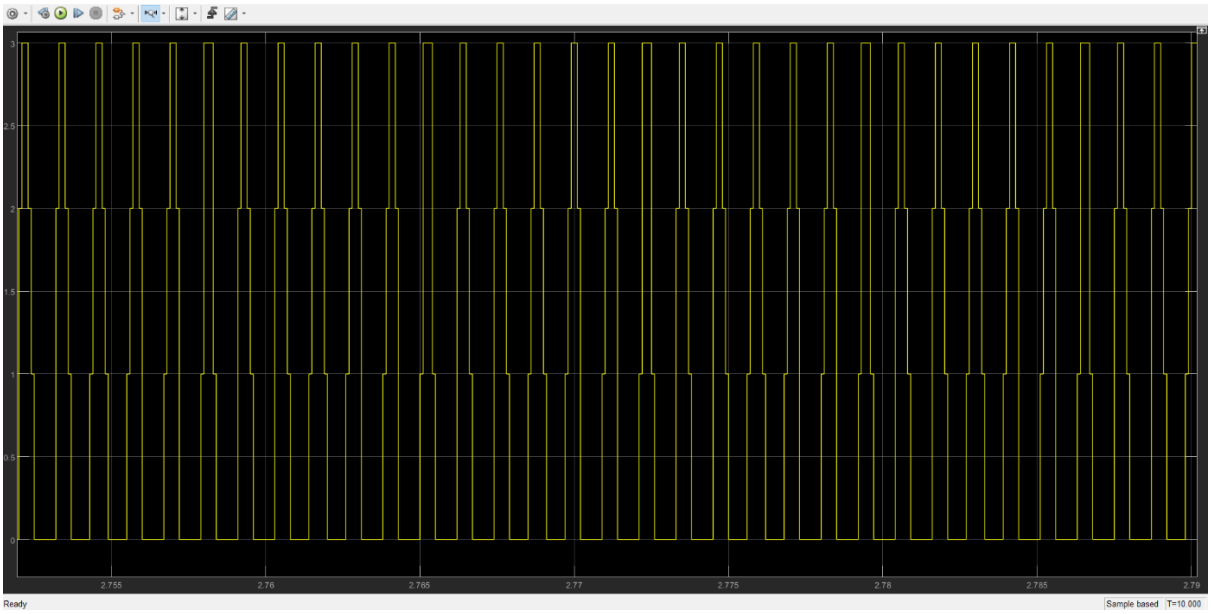

Figure 05: Output from Quantizer

- Output shows a stepped waveform, representing the quantized version of the input sine wave. This output demonstrates the discretization of the continuous input signal into distinct levels. The quantization process introduces some distortion to the original sine wave, visible as the stepped nature of the waveform.

3. Rayleigh SISO output

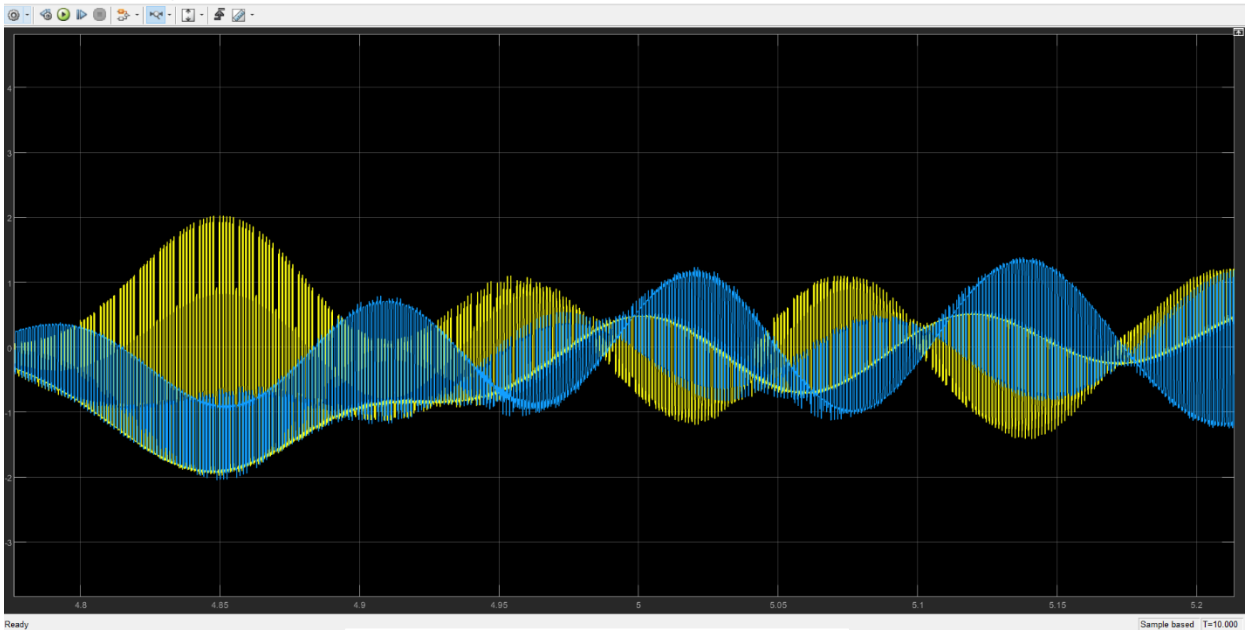

Figure 06: Output from Rayleigh SISO

- This scope output illustrates two overlapping sinusoidal waveforms with varying amplitudes. This output represents the signal before and after passing through the Rayleigh SISO channel, demonstrates the effects of multipath fading.

4. AWGN Channel output

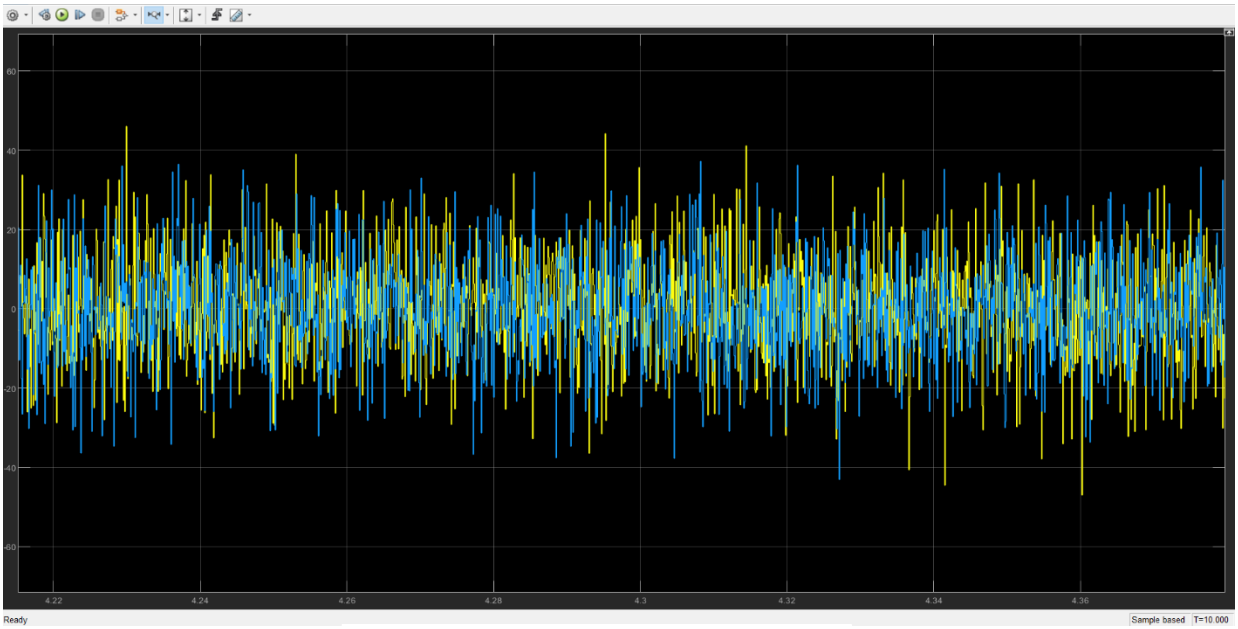

Figure 07: Output from AWGN channel

- This scope output shows a noisy, random signal with both positive and negative values. This output represents the signal after passing through the AWGN channel, demonstrating the added noise to the system.

5. Feedback diversity logic output

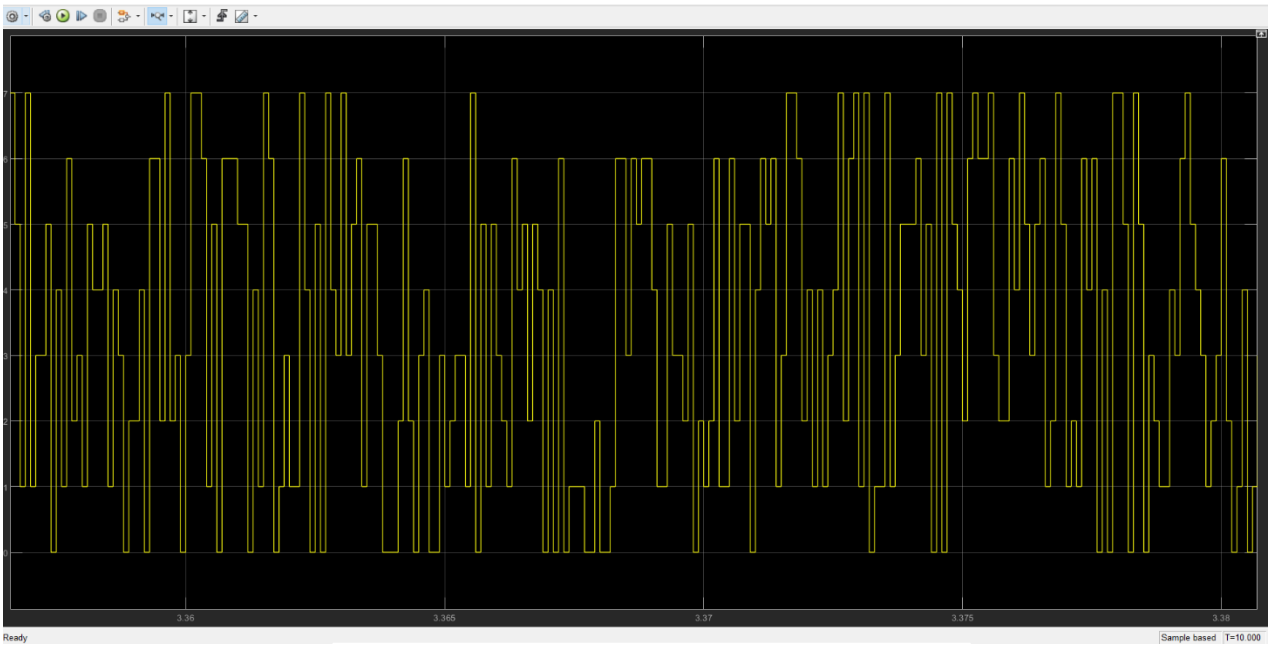

Figure 08: Output from feedback diversity logic

- This output shows a stepped waveform similar to the quantizer output, but with different characteristics. This represents the output of the feedback diversity logic, showing how the signal has been adjusted based on channel feedback.

6. Moving average filter output

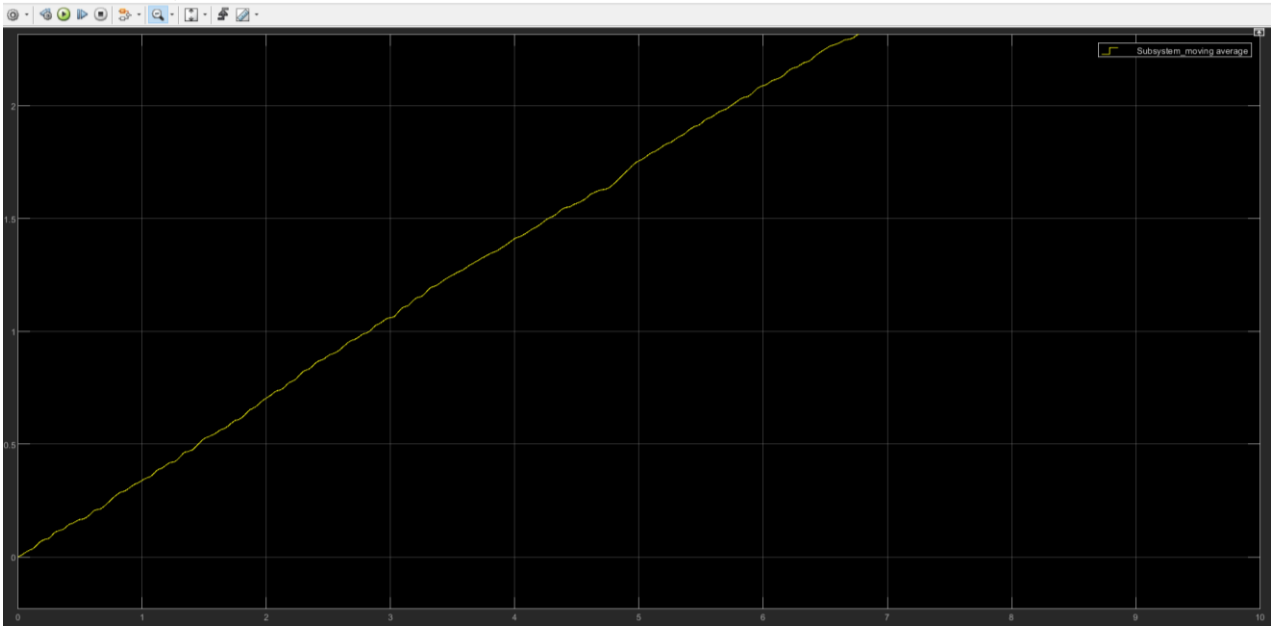

Figure 09: Output from Moving average filter

- The scope output shows increasing trend over time. The line is not perfectly smooth but exhibits small fluctuations, indicating some residual high frequency components.

RESULTS USING MATLAB CODE

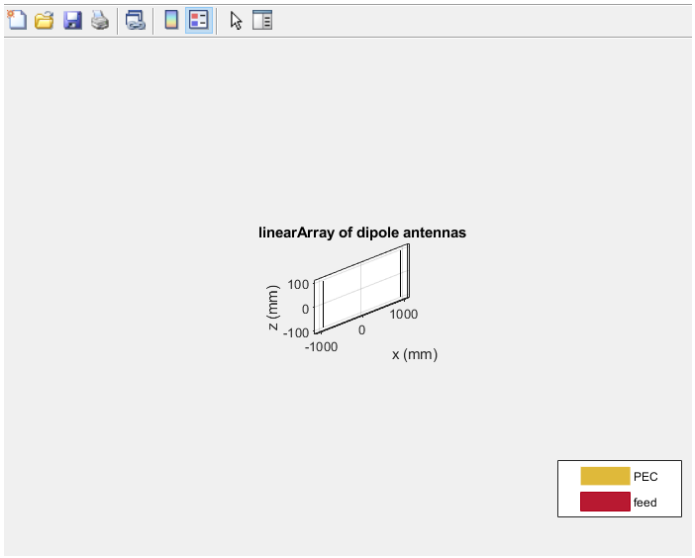

Figure 10: LinearArray of dipole antennas

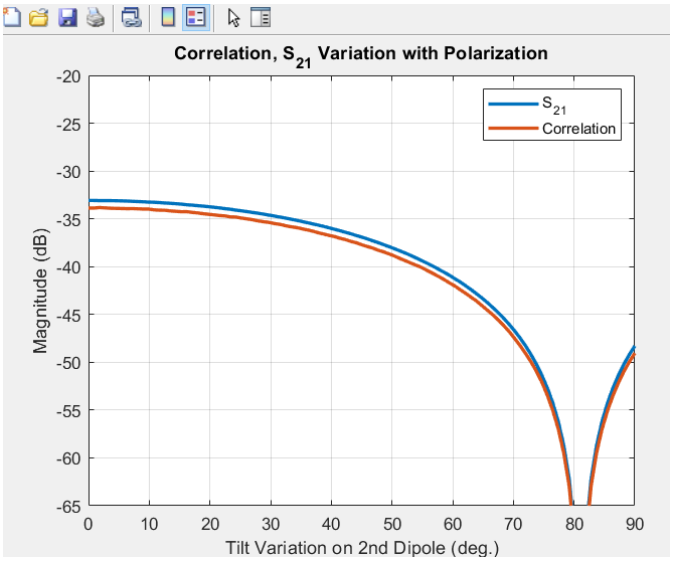

Figure 11: Correlation and S21 Vs Polarization

This image shows 3D representation of linear array of dipole antennas. This array is positioned in the x-z plane, spanning from -1000 to 1000 mm along the x-axis and -100 to 100 mm along the z-axis. This setup forms the basis of our diversity system.

Both S21 and correlation decrease as the tilt angle increases, indicating reduced coupling between antennas.

Optimal diversity is achieved at around 80 degrees tilt.

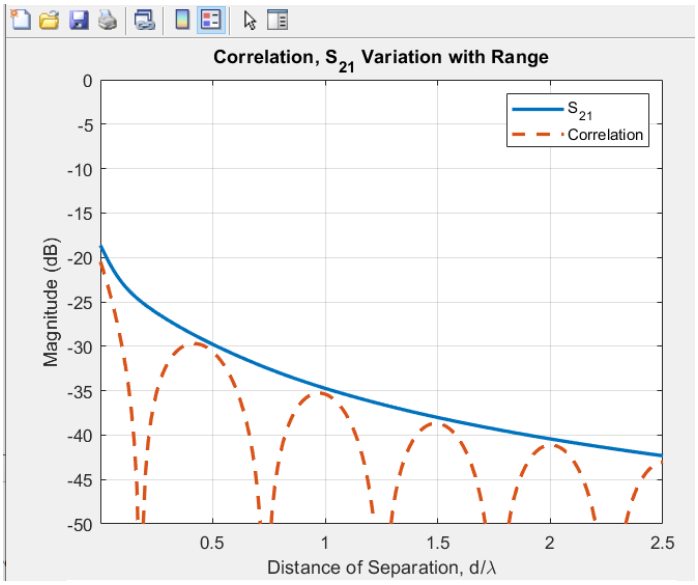

Figure 12: Correlation and S21 Vs Range

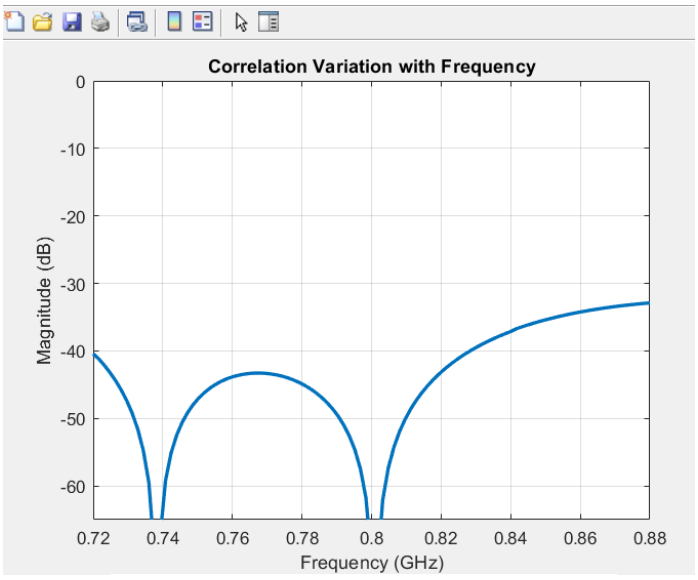

Figure 13: Correlation Vs Frequency

S21 steadily decreases with increased antenna separation.

Correlation oscillates but generally decreases with distance.

Periodic nulls in correlation indicate optimal separation distances.

Two distinct nulls at approximately 0.74 GHz and 0.8 GHz, indicating frequencies of minimal correlation.

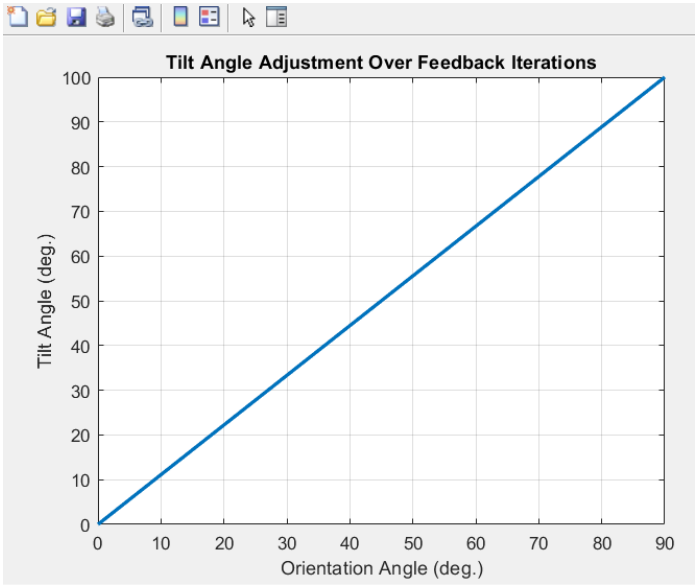

Figure 14: Tilt angle adjustment over feedback iterations

The tilt angle increases linearly from 0-100 degrees as the orientation angle increases from 0-90 degrees.

This suggests that our feedback system is adjusting the tilt angle proportionally to changes in orientation to maintain optimal diversity

5.DISCUSSION AND CONCLUSION

The feedback diversity system presented in this report leverages the spatial diversity of wireless channel to improve the overall system performance. By continuously monitoring channel conditions and adaptively selecting the optimal diversity branch, the system can maintain a more reliable and efficient.

the implemented Simulink model successfully demonstrates the concept and potential benefits of feedback diversity in wireless communication systems. By incorporating realistic channel impairments such as Rayleigh fading and AWGN, model provides challenges faced in wireless communication and how feedback diversity can help mitigate these issues.

The analysis of the scope outputs presents the effects of quantization, channel impairments, and the feedback diversity mechanism of the signal. The results show that feedback diversity is capable for improving signal quality and reliability.

From MATLAB code, results demonstrate that feedback diversity is effectively optimizing antenna parameters to minimize correlation and improve diversity performance. The linear adjustment of tilt angle in response to orientation changes indicates that feedback mechanism is working as intended, continuously adapting the array configuration to maintain diversity performance as conditions change.

6.CONTRIBUTION

Task	E – Number
Implementation using MATLAB simulink	2020/E/050 2020/E/063
Implementation using MATLAB code	2020/E/034 2020/E/197
Implementation report	2020/E/034 2020/E/050 2020/E/063 2020/E/197