Electronic and Telecommunication Engineering
University of Moratuwa, Sri Lanka

EN4384 - Wireless and Mobile Communications

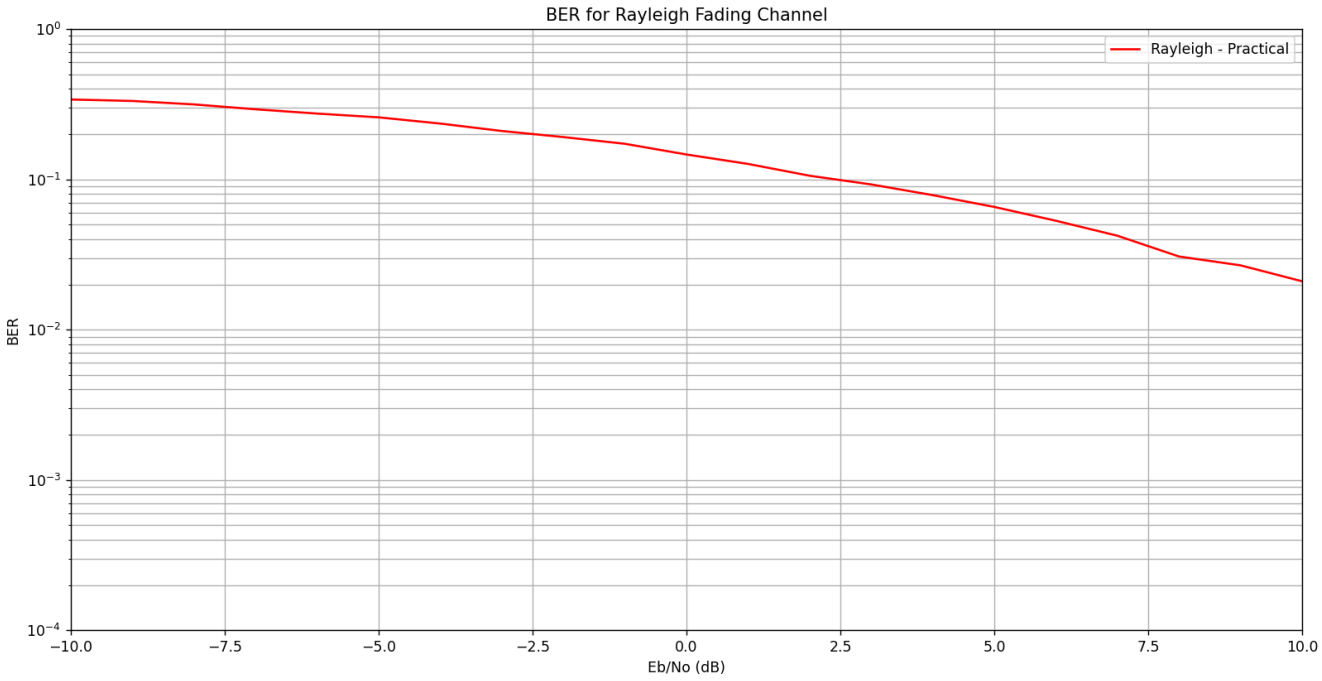
Simulation Assignment

A.A.W.L.R.Amarasinghe 210031H

This report is submitted as a partial fulfillment of module EN4384
2025.11.25

Task 1 - Average bit error rate (BER) evaluation in fading channels

a) The following plot shows the practical simulation result for the average BER of a binary phase shift keying (BPSK) in a Rayleigh fading channel. The results are plotted for $\frac{E_b}{N_0}$ values from -10 dB to +10 dB, where E_b is the bit energy and N_0 is the power spectral density of additive white Gaussian noise.



b) A theoretical expression for the BER of BPSK in Rayleigh fading channels can be derived as follows:

Let's consider the following fading channel,

$$y = hs + n$$

where y = received signal, s = transmitted signal

n = gaussian noise (AWGN)

$$P_e = AQ\left(\sqrt{b\,SNR}\right)$$

for BPSK, $b = 2$

$$P_e = AQ\left(\sqrt{2\,SNR}\right)$$

$$SNR = \frac{\text{Signal power}}{\text{noise power}}$$

$$SNR = \frac{|h|^2 |s|^2}{N_0 B}$$

Since $|s|^2 = E_b R_b$

$$SNR = \frac{|h|^2 E_b R_b}{N_0 B}$$

for raised cosine $\frac{R_b}{B} = 1$

$$SNR = \frac{|h|^2 E_b}{N_0}$$

$$\gamma = \frac{|h|^2 E_b}{N_0}$$

average BER of BPSK = $E\left[Q\left(\sqrt{2\gamma}\right)\right]$

for a rayleigh fading channel, $\beta = |h|^2$

$$f_\beta(\beta) = \frac{1}{\bar{\beta}} e^{\frac{-\beta}{\bar{\beta}}}$$

Let's take $\bar{\beta} = 1$ $\quad f_\beta(\beta) = e^{-\beta}$

$$E\left[Q\left(\sqrt{2\gamma}\right)\right] = E\left[Q\left(\sqrt{2\beta\rho}\right)\right] \qquad \rho = \frac{E_b}{N_0}$$

$$= \int_0^\infty Q\left(\sqrt{2\beta\rho}\right) f_\beta(\beta)\, d\beta$$

$$= \int_0^\infty \left[\int_{\sqrt{2\beta\rho}}^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\, dx\right] e^{-\beta}\, d\beta$$

$$= \frac{1}{\sqrt{2\pi}} \int_0^\infty \int_{\sqrt{2\beta\rho}}^\infty e^{-x^2/2} e^{-\beta}\, dx\, d\beta$$

Substitute $\frac{x}{\sqrt{2\beta\rho}} = t$

$$dx = \sqrt{2\beta\rho}\, dt$$

$$E\left[Q\left(\sqrt{2\gamma}\right)\right] = \frac{1}{\sqrt{2\pi}} \int_0^\infty \int_1^\infty e^{-\beta\rho t^2} \sqrt{2\beta\rho}\, dt\, e^{-\beta}\, d\beta$$

$$= \frac{1}{\sqrt{\pi}} \int_1^\infty \underbrace{\int_0^\infty \sqrt{\beta\rho}\, e^{-\beta\rho\left(t^2+\frac{1}{\rho}\right)} d\beta\, dt}_{M}$$

$$M = \int_0^\infty \sqrt{\beta\rho}\, e^{-\beta\rho\left(t^2+\frac{1}{\rho}\right)}\, d\beta$$

$$a = t^2 + \frac{1}{\rho} \qquad a > 0$$

$$M = \int_0^\infty \sqrt{\beta\rho}\, e^{-\beta\rho a}\, d\beta$$

Substitute $\beta\rho a = b$

$$\rho a\, d\beta = db$$

$$d\beta = \frac{1}{\rho a}\, db$$

$$M = \int_0^\infty \sqrt{\frac{b}{a}}\, e^{-b} \cdot \frac{db}{a\rho}$$

$$M = \frac{1}{a^{3/2}\rho} \underbrace{\int_0^\infty \sqrt{b}\, e^{-b}\, db}_{\text{implying gamma function}}$$

$$M = \frac{1}{a^{3/2}\rho}\,\Gamma(3/2) \qquad \Gamma(3/2) = \frac{\sqrt{\pi}}{2}$$

$$M = \frac{\sqrt{\pi}}{2a^{3/2}\rho}$$

$$E\left[Q\left(\sqrt{2\gamma}\right)\right] = \frac{1}{\sqrt{\pi}} \int_1^\infty \frac{\sqrt{\pi}}{2a^{3/2}\rho}\, dt$$

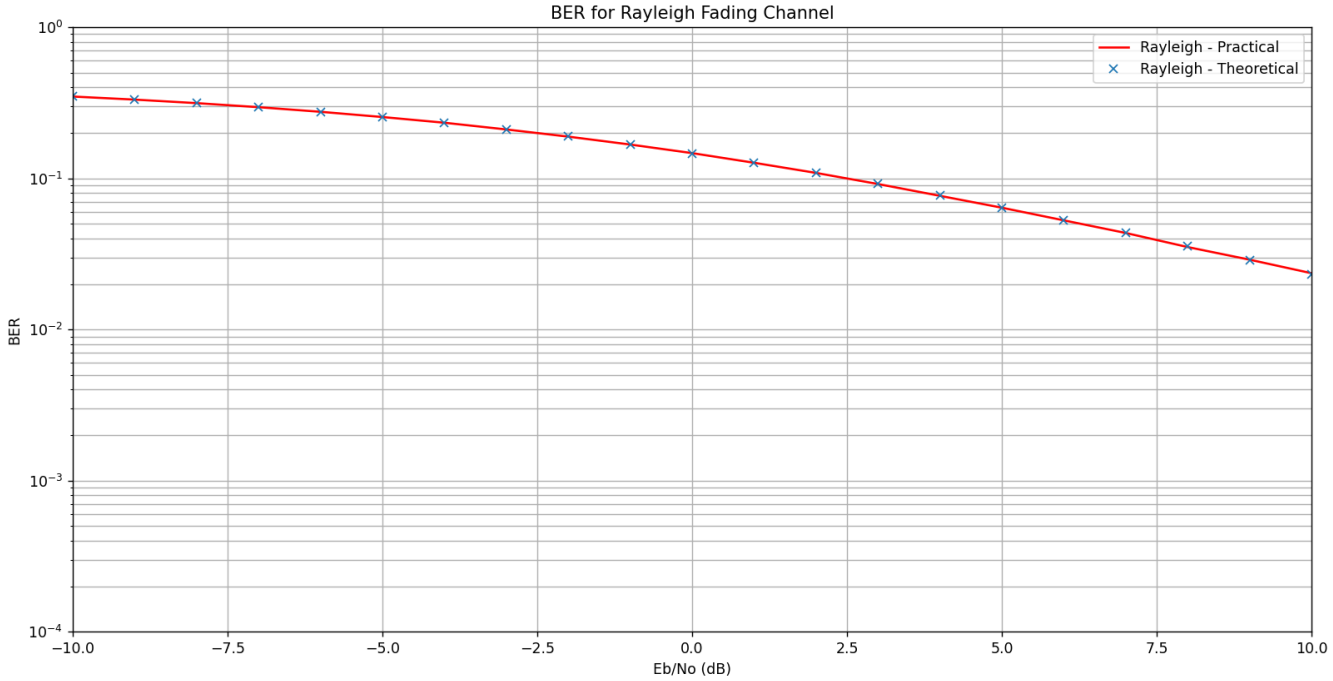
$$= \frac{1}{2\rho} \int_1^\infty \frac{1}{\left(t^2 + \frac{1}{\rho}\right)^{3/2}}\, dt$$

$$= \frac{1}{2\rho}\left[\rho - \frac{\rho^{3/2}}{(1+\rho)^{1/2}}\right]$$

$$= \frac{1}{2}\left[\rho - \sqrt{\frac{\rho}{1+\rho}}\right]$$

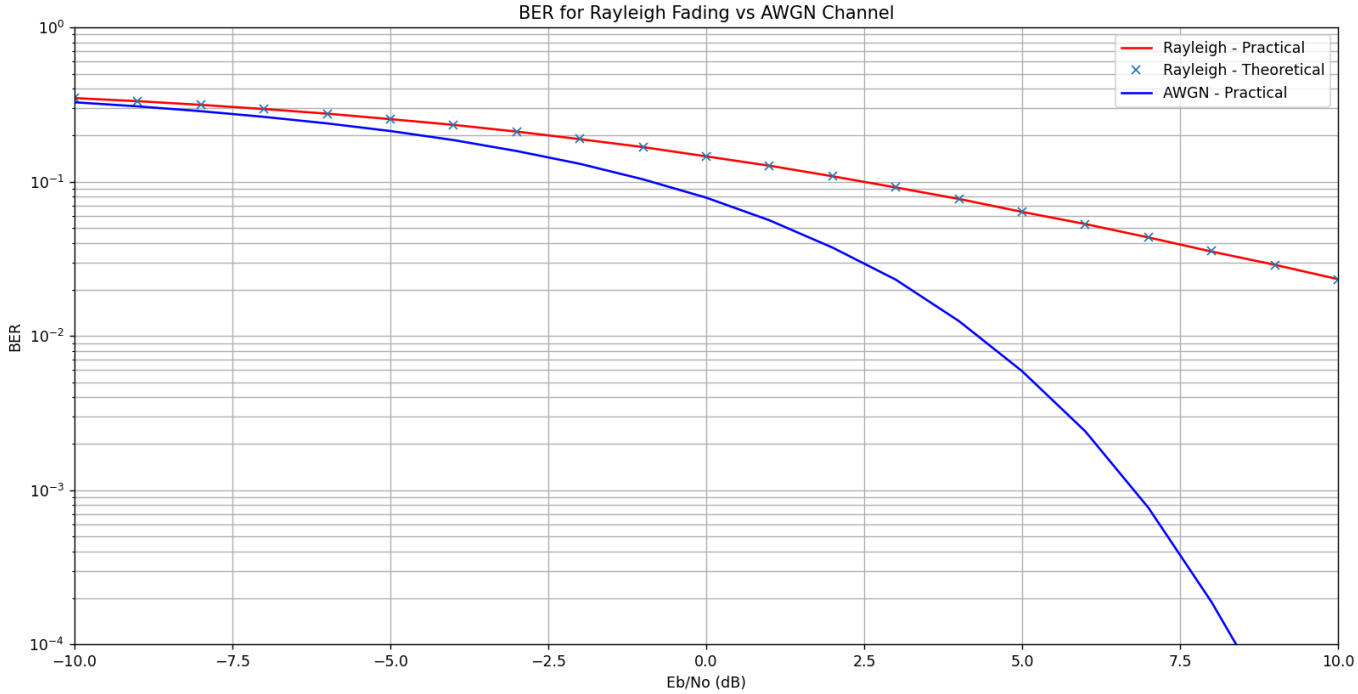
$$= \frac{1}{2}\left[1 - \sqrt{\frac{(E_b/N_o)}{1+(E_b/N_o)}}\right]$$

c) The simulation for the theoretical values and the practical values can be viewed in the same plot as follows:



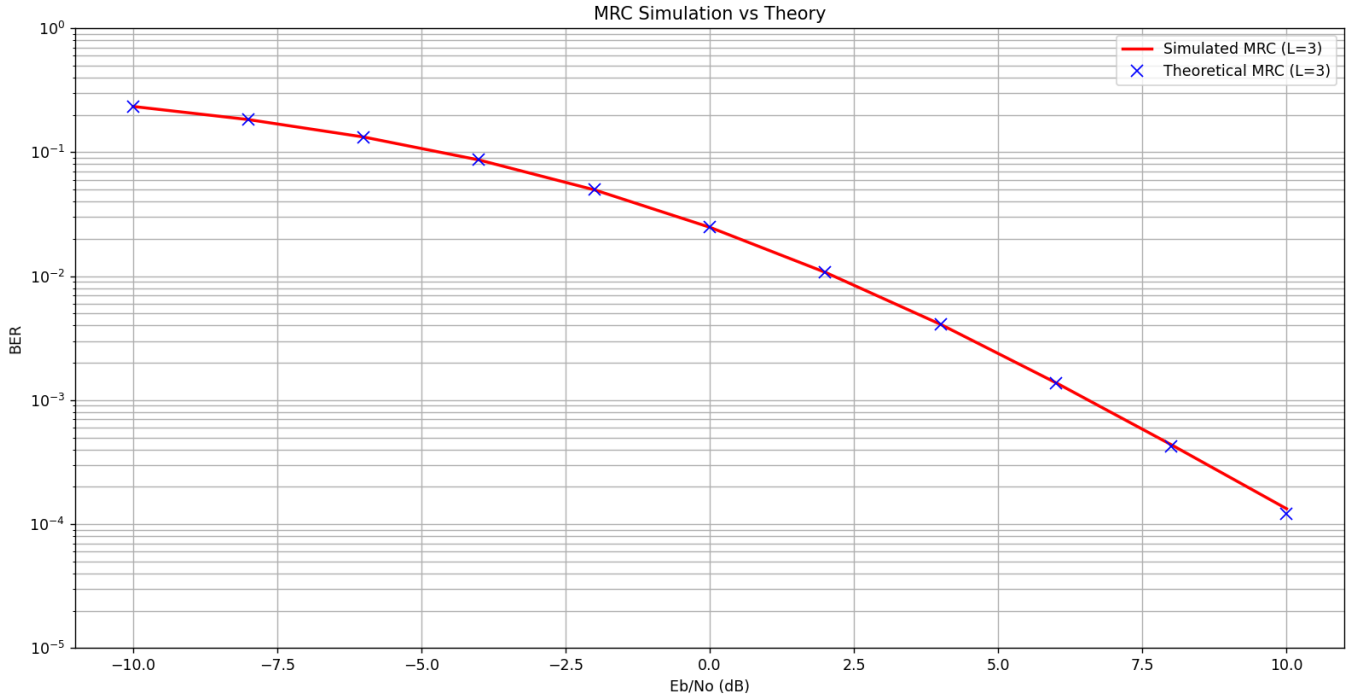
d) The Bit Error Rate (BER) decreases exponentially as the $\frac{E_b}{N_0}$ (Signal-to-Noise Ratio, SNR) increases. This trend is common for both Rayleigh fading and AWGN channels, indicating that higher $\frac{E_b}{N_0}$ leads to better performance in terms of reduced bit errors.

However, the Rayleigh fading channel shows a higher BER compared to the AWGN channel for the same values of $\frac{E_b}{N_0}$. This is due to the fading effects in Rayleigh fading channels, which cause rapid fluctuations in the signal strength, leading to more errors. In contrast, an AWGN channel, which is idealized and assumes constant signal strength, exhibits a lower BER for the same SNR values.



Task 2 – Comparing Diversity Combining Schemes

a) This plot shows both theoretical and simulated average BER of BPSK in Rayleigh fading channel using 3-branch maximal ratio combining (MRC) receiver.



b) A theoretical expression for the average BER for BPSK when using 3-channel maximum ratio combiner can be derived as follows:

For MRC, the total SNR at the combiner output is,

$$\gamma = \sum_{i=1}^{L} \gamma_i$$

where each branch SNR γ_i is exponentially distributed with mean $\bar{\gamma}$

$$P_{\gamma_i}(\gamma_i) = \frac{1}{\bar{\gamma}} e^{\frac{-\gamma_i}{\bar{\gamma}}} \quad ; \gamma_i \geq 0$$

The sum of L iid exponential random variable is gamma distributed. So,

$$P_e(\gamma) = \frac{1}{(L-1)!\,\bar{\gamma}^L} \gamma^{L-1} e^{-\gamma/\bar{\gamma}} \quad ; \gamma \geq 0$$

For BPSK in AWGN with instantaneous SNR γ, the conditional BER is,

$$P_e(\gamma) = Q(\sqrt{2\gamma})$$

Hence the average BER with MRC is

$$P_e = \int_0^\infty Q(\sqrt{2\gamma})\, P_\gamma(\gamma)\, d\gamma$$

$$P_e = \int_0^\infty Q(\sqrt{2\gamma})\, \frac{\gamma^{L-1}}{(L-1)!\,\bar{\gamma}^L}\, e^{-\gamma/\bar{\gamma}}\, d\gamma$$

Using Craig's representation of the Q function

$$Q(x) = \frac{1}{\pi} \int_0^{\pi/2} \exp\left(\frac{-x^2}{2\sin^2\theta}\right) d\theta$$

we obtain

$$Q(\sqrt{2\gamma}) = \frac{1}{\pi} \int_0^{\pi/2} \exp\left(\frac{-\gamma}{2\sin^2\theta}\right) d\theta$$

substitute this in average BER

$$P_e = \frac{1}{\pi} \int_0^{\pi/2} \int_0^\infty e^{\frac{-\gamma}{\sin^2\theta}}\, \frac{\gamma^{L-1}}{(L-1)!\,\bar{\gamma}^L} \cdot e^{-\gamma/\bar{\gamma}}\, d\gamma\, d\theta$$

$$P_e = \frac{1}{\pi\,(L-1)!\,\bar{\gamma}^L} \int_0^{\pi/2} \left[\int_0^\infty \gamma^{L-1} e^{-\gamma\left(\frac{1}{\bar{\gamma}} + \frac{1}{\sin^2\theta}\right)} d\gamma \right] d\theta$$

using standard gamma integral,

$$\int_0^\infty \gamma^{L-1} e^{-b\gamma}\, d\gamma = \frac{(L-1)!}{b^L} \quad ; \; b > 0$$

with $b = \frac{1}{\bar{\gamma}} + \frac{1}{\sin^2\theta}$ we get

$$P_e = \frac{1}{\pi \bar{\gamma}^L} \int_0^{\pi/2} \left(\frac{1}{\frac{1}{\bar{\gamma}} + \frac{1}{\sin^2\theta}} \right)^L d\theta$$

$$P_e = \frac{1}{\pi \bar{\gamma}^L} \int_0^{\pi/2} \left(\frac{\bar{\gamma} \sin^2\theta}{\bar{\gamma} + \sin^2\theta} \right)^L d\theta$$

$$P_e = \frac{1}{\pi} \int_0^{\pi/2} \left(\frac{1}{1 + \frac{\bar{\gamma}}{\sin^2\theta}} \right)^L d\theta$$

After evaluting the integral closed form, algebra we obtain the finite sum expression

$$P_e = \rho^L \sum_{k=0}^{L-1} {}^{L-1+k}C_k \, (1-\rho)^k$$

$$\rho = \frac{1}{2}\left(1 - \sqrt{\frac{\bar{\gamma}}{1+\bar{\gamma}}}\right)$$

$$\rho = \frac{1}{2}(1 - a) \quad ; \quad a = \sqrt{\frac{\bar{\gamma}}{1+\bar{\gamma}}}$$

$$P_e = p^3 \sum_{k=0}^{2} {}^{k+2}C_k \, (1-p)^k \qquad ; L = 3$$

$$P_e = p^3 \left[{}^{2}C_0 + {}^{3}C_1 (1-p) + {}^{4}C_2 (1-p)^2 \right]$$

$$= p^3 (10 - 15p + 6p^2)$$

$$= 10p^3 - 15p^4 + 6p^5$$

$$= 10 \left(\frac{1-a}{2}\right)^3 - 15 \left(\frac{1-a}{2}\right)^4 + 6 \left(\frac{1-a}{2}\right)^5$$

$$= \frac{5}{4}(1-a)^3 - \frac{15}{16}(1-a)^4 + \frac{3}{16}(1-a)^5$$

$$= (1-a)^3 \left[\frac{5}{4} - \frac{15}{16}(1-a) + \frac{3}{16}(1-2a+a^2) \right]$$

$$= (1-a)^3 \left[\frac{5}{4} - \frac{15}{16} + \frac{15a}{16} + \frac{3}{16} - \frac{3a}{8} + \frac{3a^2}{16} \right]$$

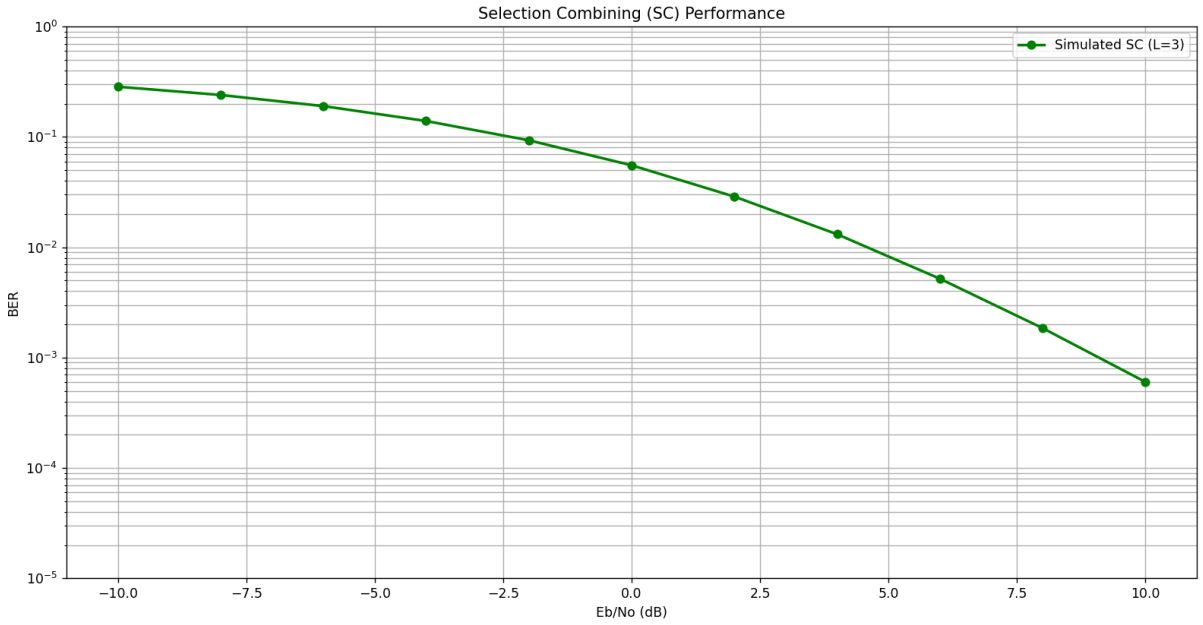
$$= (1-a)^3 \left[\frac{1}{2} + \frac{9a}{16} + \frac{3a^2}{16} \right]$$

$$= (1-3a+3a^2-a^3) \left(\frac{1}{2} + \frac{9a}{16} + \frac{3a^2}{16} \right)$$

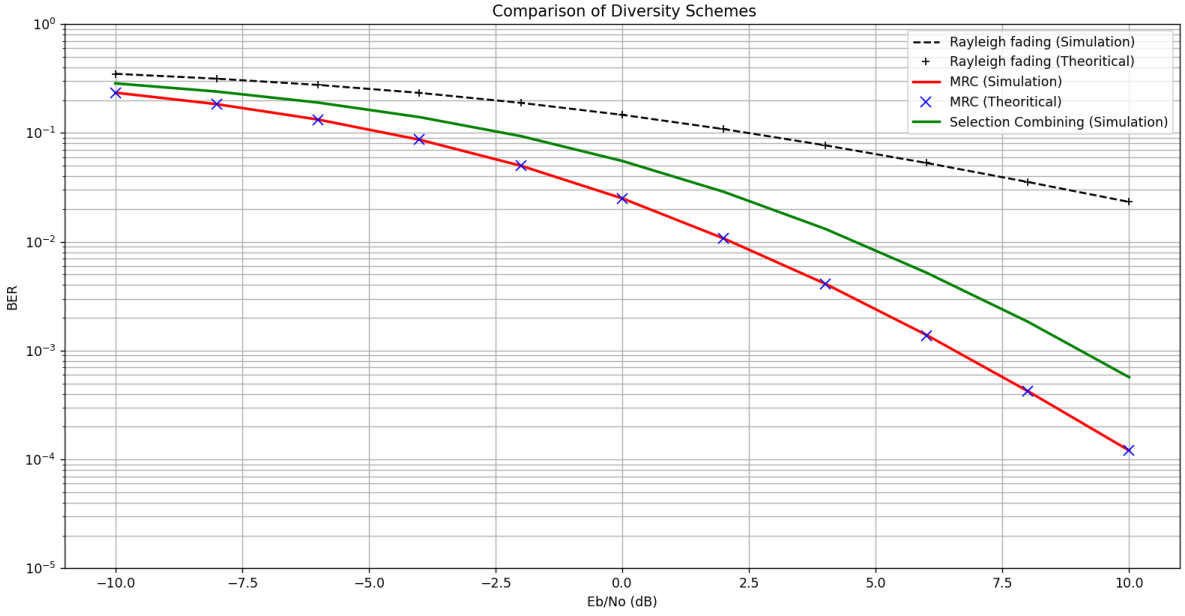
$$= \frac{1}{2} - \frac{15a}{16} + \frac{5a^3}{8} - \frac{3a^5}{16}$$

$$= \frac{1}{2} - \frac{15}{16} \left(\frac{\bar{\gamma}}{\bar{\gamma}+1}\right)^{1/2} + \frac{5}{8} \left(\frac{\bar{\gamma}}{\bar{\gamma}+1}\right)^{3/2} - \frac{3}{16} \left(\frac{\bar{\gamma}}{\bar{\gamma}+1}\right)^{5/2}$$

c) The same process is repeated using the selection combining receiver.



d) We plot the average BER for BPSK with all the techniques implemented at the receiver as follows:



e) Maximal Ratio Combining (MRC) and Selection Combining (SC) both achieve the same diversity gain. This is evident from the BER plot, where the curves for MRC and SC run parallel at high SNR values ($E_b/N_0 > 5$ dB) and show a much steeper decline compared to the Rayleigh fading case. The parallel behavior and slope indicate that both techniques achieve the full diversity order of $L = 3$, meaning they offer equal capability in mitigating deep fades and improving link reliability.

However, when considering array gain, MRC clearly outperforms SC. In the BER plot, the MRC curve is shifted to the left relative to SC, showing that for a target BER (such as 10^{-3}), MRC requires about 2 dB less E_b/N_0. This improvement occurs because MRC coherently combines the signal energy from all three branches, maximizing the effective SNR. SC, on the other hand, only uses the strongest branch and ignores the remaining two, leading to lower power efficiency and reduced array gain.

Task 3 – SVD based Decoupling of MIMO channel

The achievable rate C for a MIMO system with optimal power allocation is given by:

$$C = \sum_{i=1}^{n} \log_2 \left(1 + \frac{p_i \lambda_i}{\sigma^2}\right)$$

where:

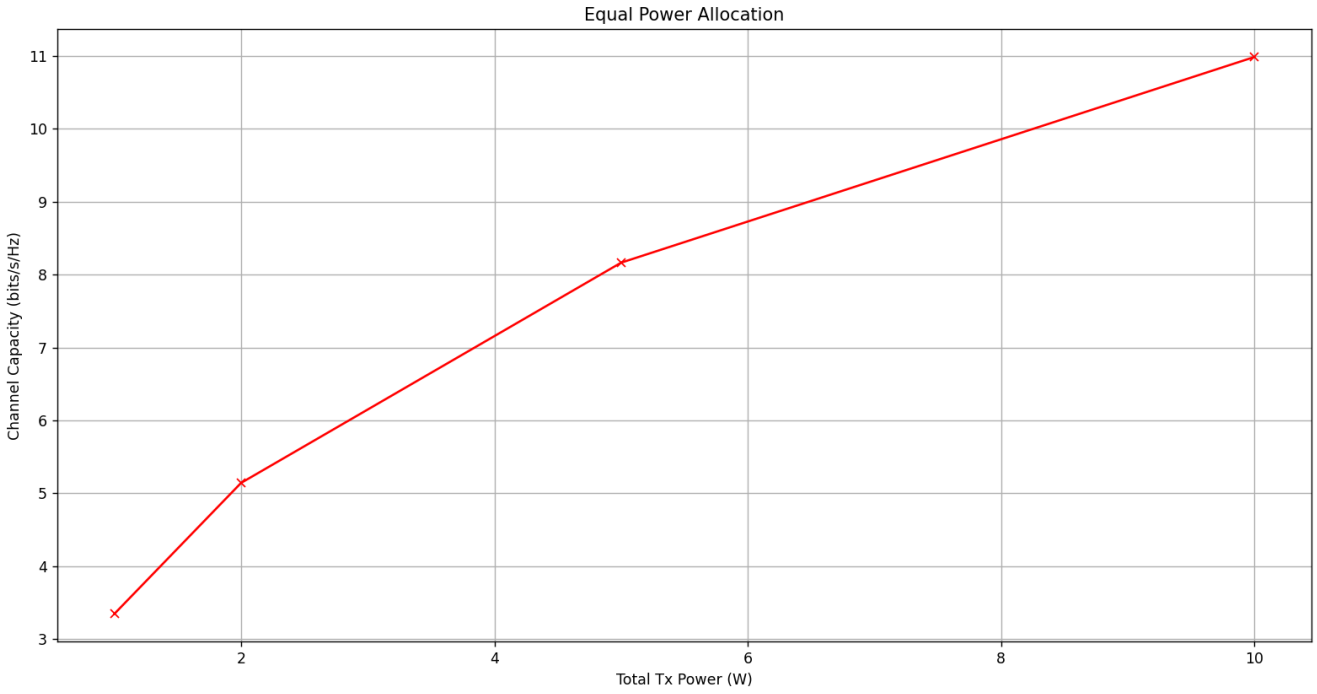
- p_i is the power allocated to the i-th eigenmode
- λ_i is the singular value (or eigenvalue) corresponding to the i-th eigenmode
- σ^2 is the noise variance at the receiver (1 in this case)
- n is the number of spatial eigenmodes (n =4 in this case)

a) Equal power allocation

In equal power allocation, the total power P_t is equally distributed across all 4 eigenmodes. So, for each eigenmode, the power $p_i = \frac{P_t}{4}$.

The achievable rate is:

$$C = \sum_{i=1}^{4} \log_2 \left(1 + \frac{P_t \lambda_i}{4}\right)$$



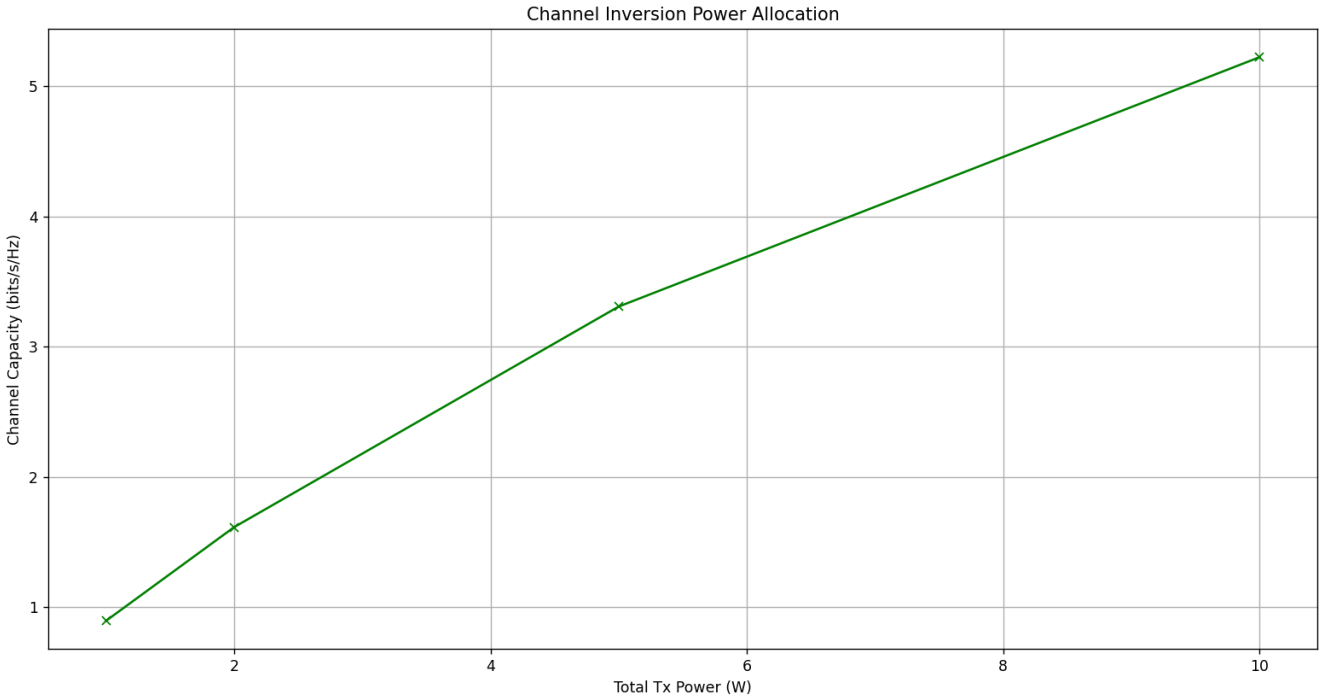
b) Channel inversion power allocation

In channel inversion, the power allocated to each eigenmode is inversely proportional to the singular value λ_i. That is:

$$p_i = \frac{1}{\lambda_i}\left(\frac{P_t}{\sum_{j=1}^{M}\frac{1}{\lambda_j}}\right)$$

The achievable rate is:

$$C = \sum_{i=1}^{4}\log_2\left(1+\frac{P_t}{\sum_{j=1}^{M}\frac{1}{\lambda_j}}\right)$$

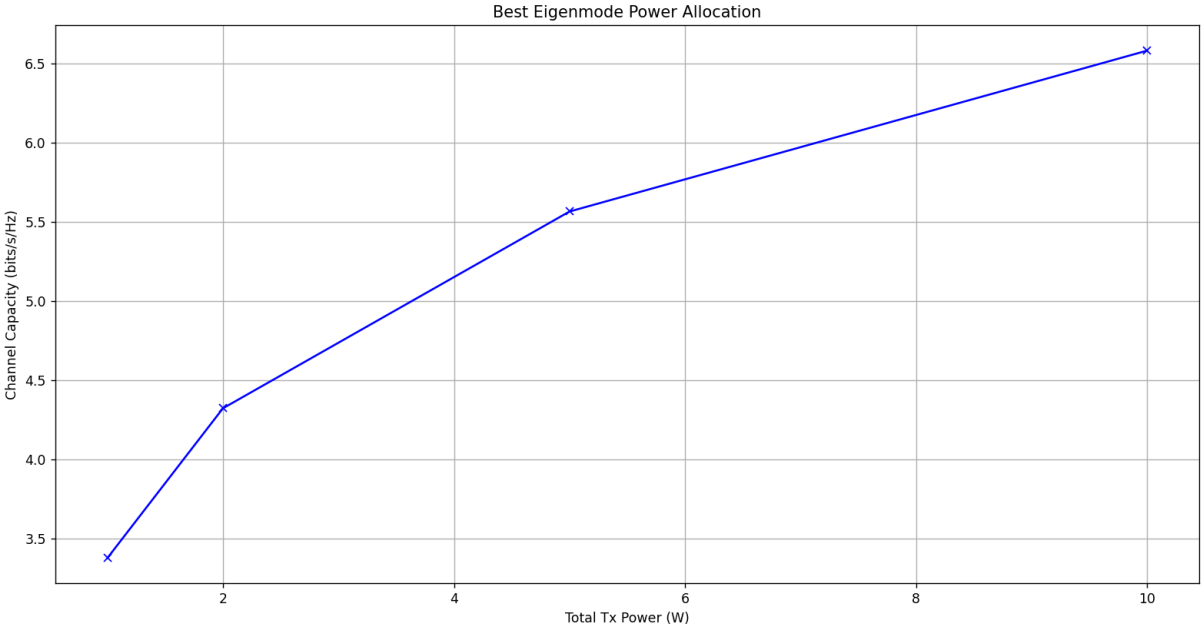


c) Allocating all power to the best eigen mode

In this case, all the power P_t is allocated to the eigenmode with the largest singular value $\lambda_{\max}$. The achievable rate is:

$$C = \log_2\left(1 + P_t\lambda_{\max}\right)$$

where $\lambda_{\max}$ is the maximum singular value.



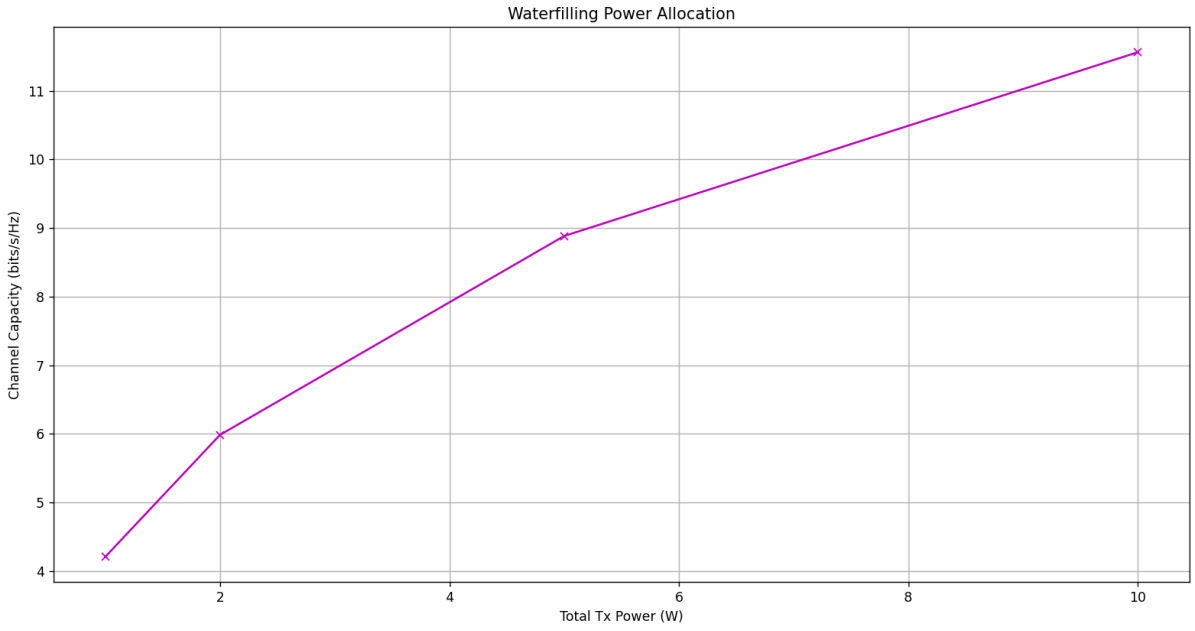
d) Water-filling power allocation

The waterfilling algorithm optimally allocates power to each eigenmode in the MIMO system. The equation for the power allocation is derived from the inverse eigenvalues and the waterfilling condition.

$$\frac{1}{\lambda} = \frac{1}{r}\left(P_t + \sum_{k=1}^{4} \frac{1}{\alpha_k}\right)$$

The power allocated to each eigenmode k is:

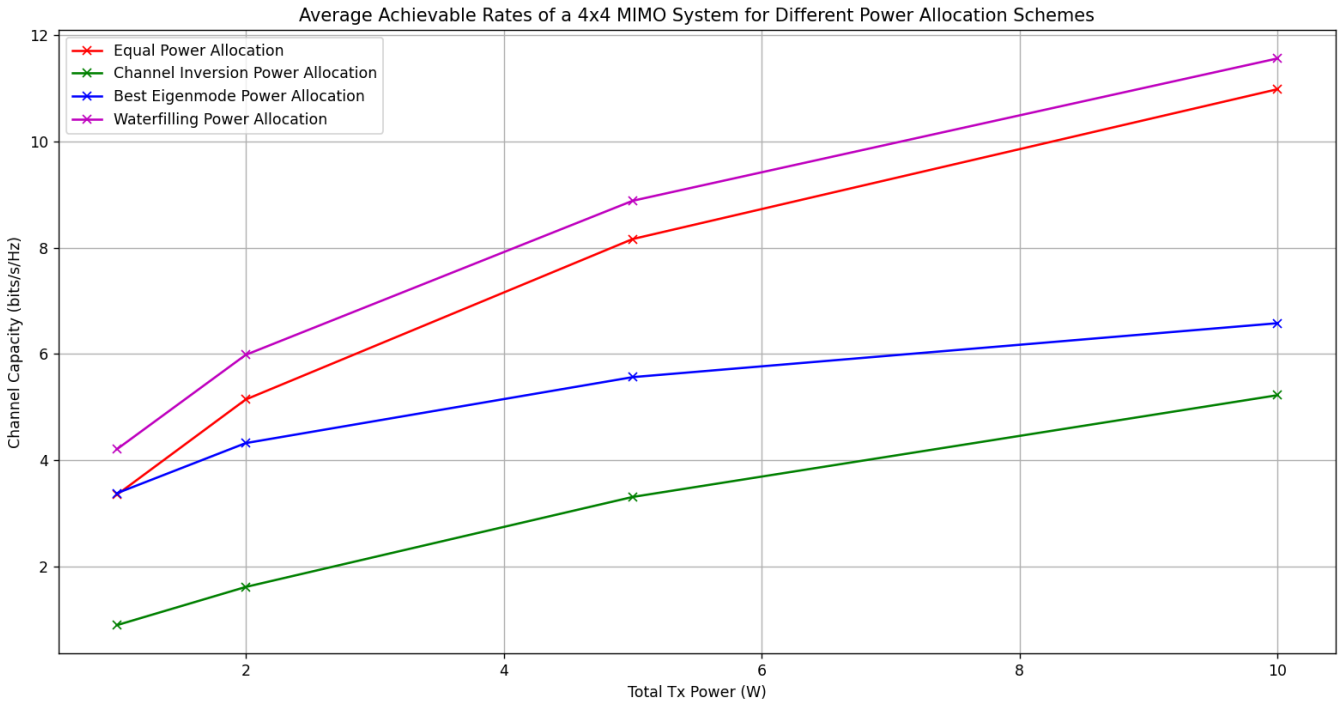
$$P_k = \frac{1}{\lambda} - \frac{1}{\alpha_k}$$



Achievable Channel Capacities for Different Power Allocation Schemes

Transmit Power (W)	**Equal Power Allocation**	**Channel Inversion Power Allocation**	**Best Eigenmode Power Allocation**	**Waterfilling Power Allocation**
1 W	3.324268	0.910852	3.372644	4.191810
2 W	5.158614	1.674251	4.312871	5.994006
5 W	8.134681	3.330218	5.573507	8.851290
10 W	11.016773	5.314778	6.569289	11.583864

Average achievable rates of a 4x4 MIMO system for all power allocation schemes are as follows:



Appendix – Python Implementation

Task 1

```python
import numpy as np
import matplotlib.pyplot as plt

# Parameters
num_points = 1000000  # Number of signal points
EbNo_dB = np.arange(-10, 11, 1)  # Eb/No in dB
Rayleigh = 0.5  # Rayleigh fading factor

# Symbol energy for Rayleigh fading (assuming noise spectral density)
symbol_energy = 10 ** (0.1 * EbNo_dB)

# Initialize BER arrays for simulation
BER_fading_array = []
BER_fading_array_theoretical = []
BER_awgn_array = []

# Simulate Rayleigh fading channel and AWGN for comparison
for Es in symbol_energy:
    # Generate bit stream
    bit_stream = np.random.randint(0, 2, num_points)
    # Map bits to symbols (BPSK modulation)
    symbol = np.sqrt(Es) * (2 * bit_stream - 1)

    # Rayleigh fading channel simulation
    h = np.sqrt(Rayleigh) * (np.random.randn(num_points) + 1j * np.random.randn(num_points))

    # Add complex Gaussian noise to the signal
    n = np.sqrt(Rayleigh) * (np.random.randn(num_points) + 1j * np.random.randn(num_points))

    # Received signal in Rayleigh fading channel
    y = symbol * h + n
    z = np.conj(h) * y  # Conjugate of the channel for coherent detection

    # Demodulate received signal
    output_decoded = np.real(z) > 0  # BPSK demodulation

    # Calculate practical BER for Rayleigh fading
    BER_fading_array.append(np.mean(np.abs(output_decoded - bit_stream)))

    # Theoretical BER for Rayleigh fading
    BER_fading_array_theoretical.append(0.5 * (1 - np.sqrt(Es / (Es + 1))))

    # ---- AWGN Channel Simulation for comparison ----
    # AWGN noise
    noise = np.sqrt(0.5) * (np.random.randn(num_points) + 1j * np.random.randn(num_points))

    # Received signal in AWGN
```

```
    y_awgn = symbol + noise

    # Demodulate received signal in AWGN
    output_decoded_awgn = np.real(y_awgn) > 0

    # Calculate BER for AWGN
    BER_awgn_array.append(np.mean(np.abs(output_decoded_awgn - bit_stream)
       ))

# Plotting the results including AWGN
plt.figure(figsize=(20, 10))
plt.semilogy(EbNo_dB, BER_fading_array, 'r', label='Rayleigh - Practical')
     # Simulated Rayleigh Fading
plt.semilogy(EbNo_dB, BER_fading_array_theoretical, 'x', label='Rayleigh -
    Theoretical')  # Theoretical Rayleigh
plt.semilogy(EbNo_dB, BER_awgn_array, 'b', label='AWGN - Practical')  #
   Simulated AWGN
plt.xlabel('Eb/No (dB)')
plt.ylabel('BER')
plt.grid(True, which="both")
plt.title('BER for Rayleigh Fading vs AWGN Channel')
plt.legend()
plt.axis([-10, 10, 1e-4, 1])
plt.show()
```

Task 2

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import factorial

num_samples = 100000  # Number of bits (Higher = smoother curve)
num_trials = 50       # Number of Monte Carlo loops
EbNo_range_dB = np.arange(-10, 12, 2)
signal_energy = 10**(0.1 * EbNo_range_dB)
L = 3  # Number of branches

# Initialize lists to store results
ber_rayleigh_sim = [] # Single branch
ber_mrc_sim = []      # Task 2a
ber_sc_sim = []       # Task 2c

for energy in signal_energy:
    err_single = 0
    err_mrc = 0
    err_sc = 0

    for _ in range(num_trials):
        # Transmit
        bits = np.random.randint(0, 2, num_samples)
        symbols = np.sqrt(energy) * (2 * bits - 1)
        tx_signal = np.tile(symbols, (L, 1))

        # Channel (Rayleigh Fading) & Noise
```

```
        h = np.sqrt(0.5) * (np.random.randn(L, num_samples) + 1j * np.
            random.randn(L, num_samples))
        n = np.sqrt(0.5) * (np.random.randn(L, num_samples) + 1j * np.
            random.randn(L, num_samples))
        r = tx_signal * h + n

        # Single Branch (Reference)
        y_single = np.conj(h[0]) * r[0]
        dec_single = np.real(y_single) > 0
        err_single += np.mean(np.abs(dec_single - bits))

        y_mrc = np.sum(np.conj(h) * r, axis=0)
        dec_mrc = np.real(y_mrc) > 0
        err_mrc += np.mean(np.abs(dec_mrc - bits))

        # Find index of branch with max magnitude
        mag = np.abs(h)
        best_idx = np.argmax(mag, axis=0)

        # Extract h and r for the best branch only
        r_sel = r[best_idx, np.arange(num_samples)]
        h_sel = h[best_idx, np.arange(num_samples)]

        y_sc = np.conj(h_sel) * r_sel
        dec_sc = np.real(y_sc) > 0
        err_sc += np.mean(np.abs(dec_sc - bits))

    # Average over trials
    ber_rayleigh_sim.append(err_single / num_trials)
    ber_mrc_sim.append(err_mrc / num_trials)
    ber_sc_sim.append(err_sc / num_trials)

# Theoretical MRC Formula
mu = np.sqrt(signal_energy / (1 + signal_energy))
ber_mrc_theory = 0
for k in range(L):
    binom = factorial(L - 1 + k) / (factorial(k) * factorial(L - 1))
    ber_mrc_theory += binom * ((1 + mu) / 2)**k
ber_mrc_theory *= ((1 - mu) / 2)**L  # Apply pre-factor

# Theoretical Single Branch Formula
ber_rayleigh_theory = 0.5 * (1 - mu)

# MRC Only
plt.figure(figsize=(8, 6))
plt.semilogy(EbNo_range_dB, ber_mrc_sim, 'r-', linewidth=2, label='
    Simulated MRC (L=3)')
plt.semilogy(EbNo_range_dB, ber_mrc_theory, 'bx', markersize=8, label='
    Theoretical MRC (L=3)')
plt.title('BER for BPSK with 3-branch Maximal Ratio Combining')
plt.xlabel('Eb/No (dB)')
plt.ylabel('BER')
plt.grid(True, which='both')
plt.legend()
plt.ylim([1e-5, 1])
```

```

# Selection Combining
plt.figure(figsize=(8, 6))
plt.semilogy(EbNo_range_dB, ber_sc_sim, 'g-o', linewidth=2, label='
    Simulated SC (L=3)')
plt.title('BER for BPSK with Selection Combining')
plt.xlabel('Eb/No (dB)')
plt.ylabel('BER')
plt.grid(True, which='both')
plt.legend()
plt.ylim([1e-5, 1])

# Comparison of All Schemes
plt.figure(figsize=(10, 6))
# 1. Single Branch (No Diversity)
plt.semilogy(EbNo_range_dB, ber_rayleigh_sim, 'k--', label='Rayleigh
    fading (Simulation)')
plt.semilogy(EbNo_range_dB, ber_rayleigh_theory, 'k+', label='Rayleigh
    fading (Theoritical)')
# 2. MRC
plt.semilogy(EbNo_range_dB, ber_mrc_sim, 'r-', linewidth=2, label='MRC (
    Simulation)')
plt.semilogy(EbNo_range_dB, ber_mrc_theory, 'bx', markersize=8, label='MRC
     (Theoritical)')
# 3. SC
plt.semilogy(EbNo_range_dB, ber_sc_sim, 'g-', linewidth=2, label='
    Selection Combining (Simulation)')
plt.title(' Comparison of Diversity Schemes')
plt.xlabel('Eb/No (dB)')
plt.ylabel('BER')
plt.grid(True, which='both')
plt.legend()
plt.ylim([1e-5, 1])

plt.show()
```

Task 3

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
mT = 4  # Number of transmitter antennas
mR = 4  # Number of receiver antennas
noise_variance = 1  # Noise variance
transmitter_power = [1, 2, 5, 10]  # Transmitter power values (W)
num_realizations = 1000
capacity_results = np.zeros((4, len(transmitter_power), num_realizations))

for j, Pt in enumerate(transmitter_power):
    for i in range(num_realizations):
        # Generating random channel matrix H
        H = np.sqrt(1 / 2) * (np.random.randn(mR, mT) + 1j * np.random.
            randn(mR, mT))

```

```
        # Singular value decomposition of H
        singular_values = np.linalg.svd(H, compute_uv=False)
        eig_vals = singular_values**2 / noise_variance

        # Equal power allocation
        power_equal = Pt / mT
        capacity_results[0, j, i] = np.sum(np.log2(1 + (power_equal *
            eig_vals)))

        # Channel inversion power allocation
        power_channel_inv = Pt / np.sum(1 / eig_vals)
        capacity_results[1, j, i] = mT * np.log2(1 + power_channel_inv)

        # Allocating all power to the best eigenmode
        capacity_results[2, j, i] = np.log2(1 + (Pt * np.max(eig_vals)))

        # Waterfilling power allocation
        WF_pow = (Pt + np.sum(1 / eig_vals)) / len(eig_vals) - 1 /
            eig_vals
        while np.any(WF_pow < 0):
            non_positive_indices = WF_pow <= 0
            positive_indices = WF_pow > 0
            remaining_eigenvalues = eig_vals[positive_indices]
            num_positive = len(remaining_eigenvalues)
            WF_pow_temp = (Pt + np.sum(1 / remaining_eigenvalues)) /
                num_positive - 1 / remaining_eigenvalues
            WF_pow[non_positive_indices] = 0
            WF_pow[positive_indices] = WF_pow_temp

        capacity_results[3, j, i] = np.sum(np.log2(1 + (WF_pow * eig_vals)
            ))

# Calculate average capacity
average_capacity = np.mean(capacity_results, axis=2)

# Plot Equal Power Allocation with crosses
plt.figure(figsize=(20, 10))
plt.plot(transmitter_power, average_capacity[0, :], 'xr-', label='Equal
    Power Allocation')  # Red with crosses
plt.xlabel('Total Tx Power (W)')
plt.ylabel('Channel Capacity (bits/s/Hz)')
plt.title('Equal Power Allocation')
plt.grid()

# Plot Channel Inversion Power Allocation with crosses
plt.figure(figsize=(20, 10))
plt.plot(transmitter_power, average_capacity[1, :], 'xg-', label='Channel
    Inversion Power Allocation')  # Green with crosses
plt.xlabel('Total Tx Power (W)')
plt.ylabel('Channel Capacity (bits/s/Hz)')
plt.title('Channel Inversion Power Allocation')
plt.grid()

# Plot Best Eigenmode Power Allocation with crosses
plt.figure(figsize=(20, 10))
```

```
plt.plot(transmitter_power, average_capacity[2, :], 'xb-', label='Best Eigenmode Power Allocation')  # Blue with crosses
plt.xlabel('Total Tx Power (W)')
plt.ylabel('Channel Capacity (bits/s/Hz)')
plt.title('Best Eigenmode Power Allocation')
plt.grid()

# Plot Waterfilling Power Allocation with crosses
plt.figure(figsize=(20, 10))
plt.plot(transmitter_power, average_capacity[3, :], 'xm-', label='Waterfilling Power Allocation')  # Magenta with crosses
plt.xlabel('Total Tx Power (W)')
plt.ylabel('Channel Capacity (bits/s/Hz)')
plt.title('Waterfilling Power Allocation')
plt.grid()

# Combine all plots into a single figure with updated markers (crosses)
plt.figure(figsize=(20, 10))
plt.plot(transmitter_power, average_capacity[0, :], 'xr-', label='Equal Power Allocation')
plt.plot(transmitter_power, average_capacity[1, :], 'xg-', label='Channel Inversion Power Allocation')
plt.plot(transmitter_power, average_capacity[2, :], 'xb-', label='Best Eigenmode Power Allocation')
plt.plot(transmitter_power, average_capacity[3, :], 'xm-', label='Waterfilling Power Allocation')
plt.xlabel('Total Tx Power (W)')
plt.ylabel('Channel Capacity (bits/s/Hz)')
plt.title('Average Achievable Rates of a 4x4 MIMO System for Different Power Allocation Schemes')
plt.legend()
plt.grid()
plt.show()

# Create a table to display the results
import pandas as pd

# Convert average_capacity to a DataFrame for easier table manipulation
table_data = np.vstack([average_capacity[0, :],
                        average_capacity[1, :],
                        average_capacity[2, :],
                        average_capacity[3, :]]).T

# Create a DataFrame with columns for each power allocation scheme
df = pd.DataFrame(table_data, columns=[
    'Equal Power Allocation',
    'Channel Inversion Power Allocation',
    'Best Eigenmode Power Allocation',
    'Waterfilling Power Allocation'],
    index=[f'{pt} W' for pt in transmitter_power])

# Display the table
print(df)
```