- # Combine of zero forcing and mmse

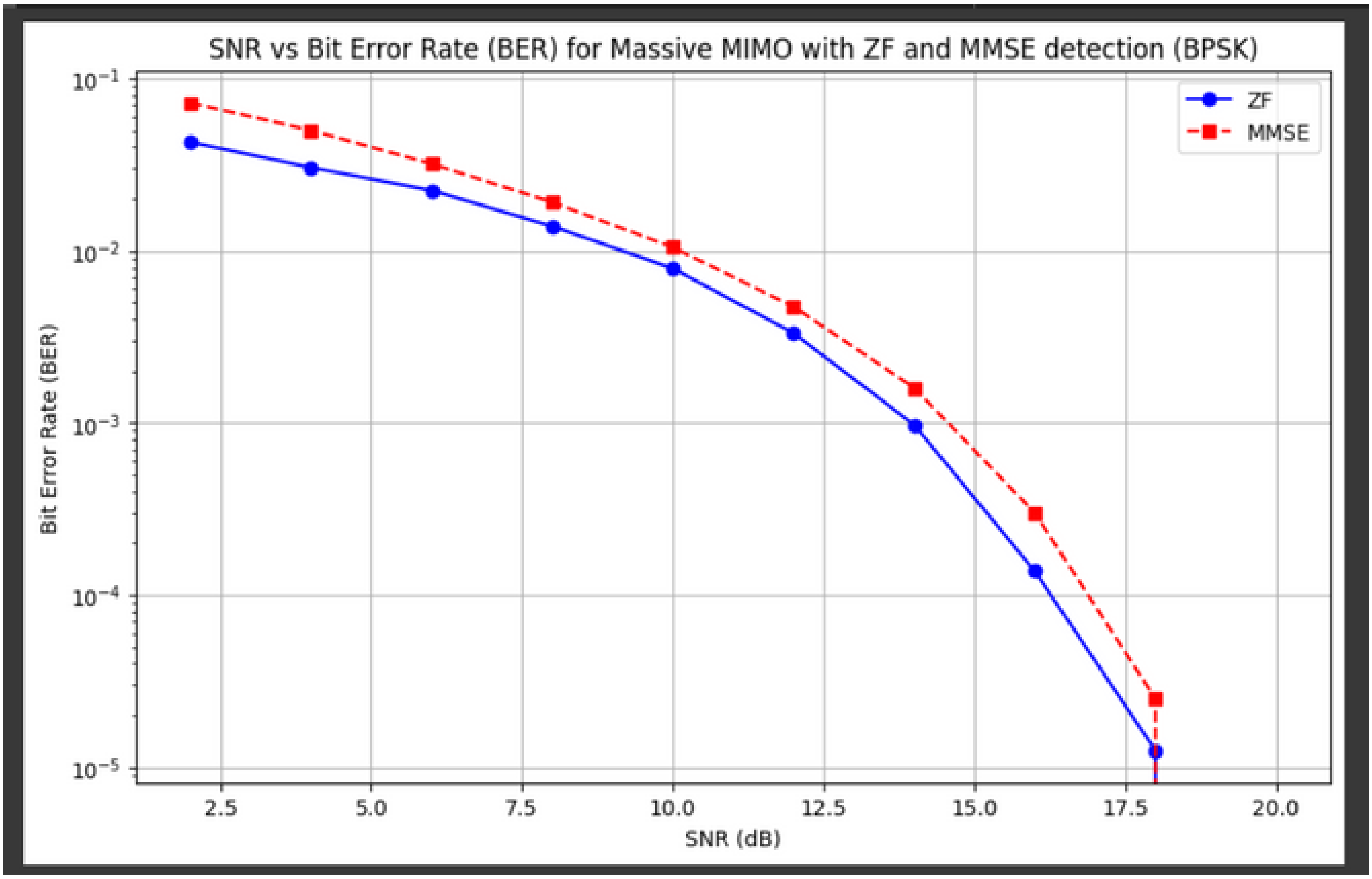
```
import numpy as np
import matplotlib.pyplot as plt

# Function to generate additive white Gaussian noise (AWGN)
def awgn_noise(signal, noise_power):
    noise = np.random.randn(*signal.shape) * np.sqrt(noise_power)
    return noise

# Function to simulate Massive MIMO system with Zero Forcing (ZF) detection
def simulate_mimo_system_zf(H, num_users, snr_values_db, num_trials=10000):
    # Generate random symbol vector x_true
    num_symbols = num_users
    x_true = np.random.randint(0, 2, num_symbols) * 2 - 1  # BPSK symbols {-1, 1}

    ber_values_zf = []

    for snr_db_val in snr_values_db:
        # Convert SNR from dB to linear scale
        snr_lin = 10**(snr_db_val / 10)
        noise_power = 1 / snr_lin

        num_errors = 0

        for _ in range(num_trials):
            # Generate received signal with AWGN
            y = np.dot(H, x_true) + awgn_noise(np.dot(H, x_true), noise_power)
```

```
            # Zero Forcing (ZF) detection
            H_pinv = np.linalg.pinv(H)
            x_demod_zf = np.sign(np.dot(H_pinv, y.real))  # Demodulate symbols

            # Calculate Bit Error Rate (BER) for ZF
            num_errors += np.sum(x_demod_zf != x_true)

        ber = num_errors / (num_trials * num_symbols)
        ber_values_zf.append(ber)

    return ber_values_zf

# Function to simulate Massive MIMO system with MMSE detection
def simulate_mimo_system_mmse(H, num_users, snr_values_db, num_trials=10000):
    # Generate random symbol vector x_true
    num_symbols = num_users
    x_true = np.random.randint(0, 2, num_symbols) * 2 - 1  # BPSK symbols {-1, 1}

    ber_values_mmse = []

    for snr_db_val in snr_values_db:
        # Convert SNR from dB to linear scale
        snr_lin = 10**(snr_db_val / 10)
        noise_power = 1 / snr_lin

        num_errors = 0

        for _ in range(num_trials):
            # Generate received signal with AWGN
            y = np.dot(H, x_true) + awgn_noise(np.dot(H, x_true), noise_power)
```

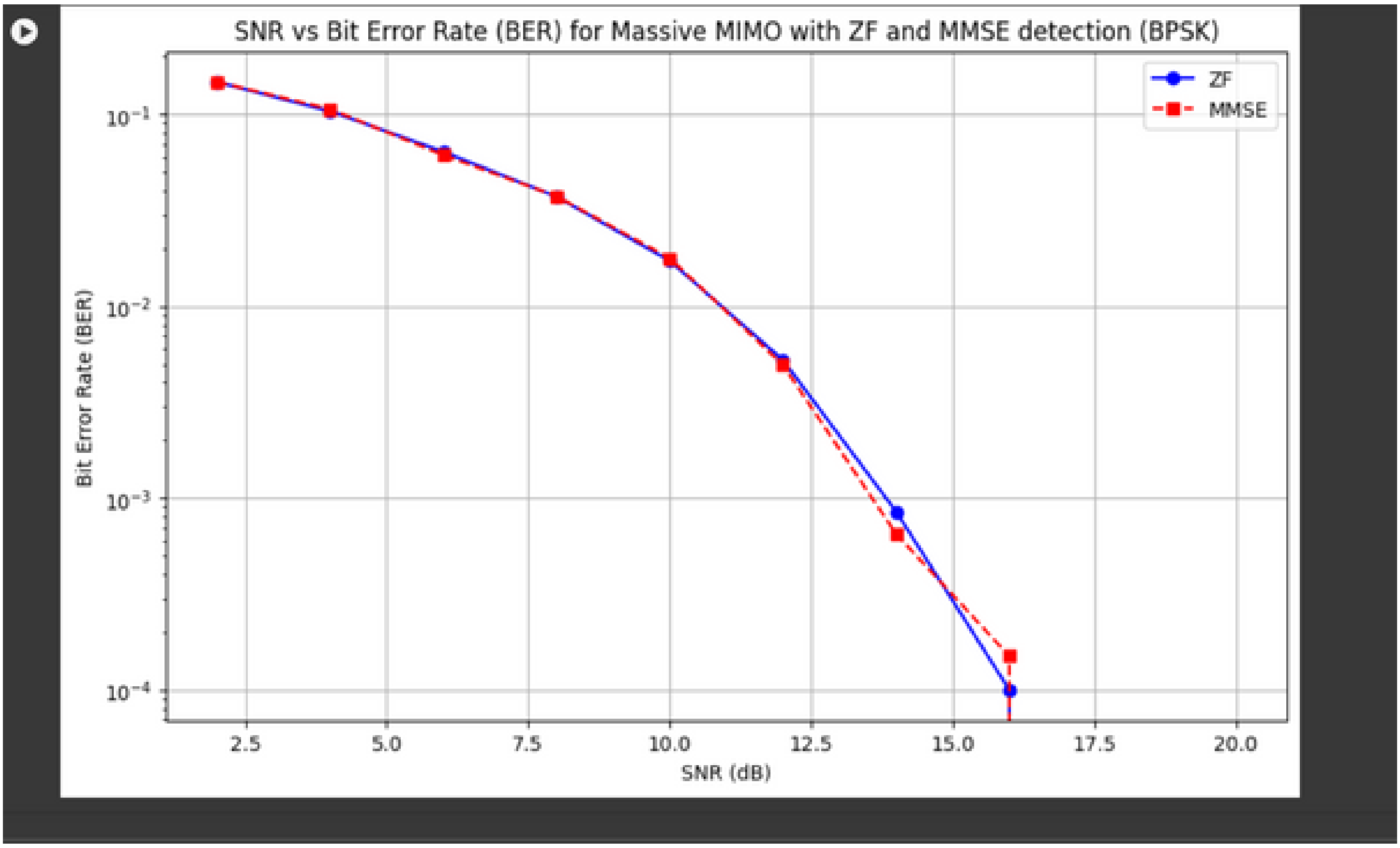
```
        # MMSE detection
        part1_w = np.conj(H.T) @ H
        part2_W = np.linalg.inv(part1_w + noise_power * np.eye(num_users))
        W_mmse = part2_W @ np.conj(H.T)
        x_demod_mmse = np.sign(W_mmse @ y.real)  # Demodulate symbols

        # Calculate Bit Error Rate (BER) for MMSE
        num_errors += np.sum(x_demod_mmse != x_true)

    ber = num_errors / (num_trials * num_symbols)
    ber_values_mmse.append(ber)

  return ber_values_mmse

# Parameters
num_antennas = 8
num_users = 4  # Change this to the desired number of users
modulation_order = 2  # BPSK modulation
num_trials = 10000

# SNR in dB (from 2 dB to 20 dB)
snr_values_db = np.arange(2, 21, 2)

# Generate random channel matrix H
H = np.random.randn(num_antennas, num_users) + 1j * np.random.randn(num_antennas, num_users)
H = H / np.sqrt(2)  # Scale every element by 1/sqrt(2)

# Simulate Massive MIMO system with Zero Forcing (ZF) detection for BPSK
ber_values_zf = simulate_mimo_system_zf(H, num_users, snr_values_db, num_trials)

# Simulate Massive MIMO system with MMSE detection for BPSK
```

```python
ber_values_mmse = simulate_mimo_system_mmse(H, num_users, snr_values_db, num_trials)


# Plot SNR vs BER for both ZF and MMSE
plt.figure(figsize=(10, 6))
plt.semilogy(snr_values_db, ber_values_zf, marker='o', linestyle='-', label='ZF', color='blue')
plt.semilogy(snr_values_db, ber_values_mmse, marker='s', linestyle='--', label='MMSE', color='red')
plt.title('SNR vs Bit Error Rate (BER) for Massive MIMO with ZF and MMSE detection (BPSK)')
plt.xlabel('SNR (dB)')
plt.ylabel('Bit Error Rate (BER)')
plt.legend()
plt.grid(True)
plt.show()
```

16X8



2X4



Zero Forcing Code :

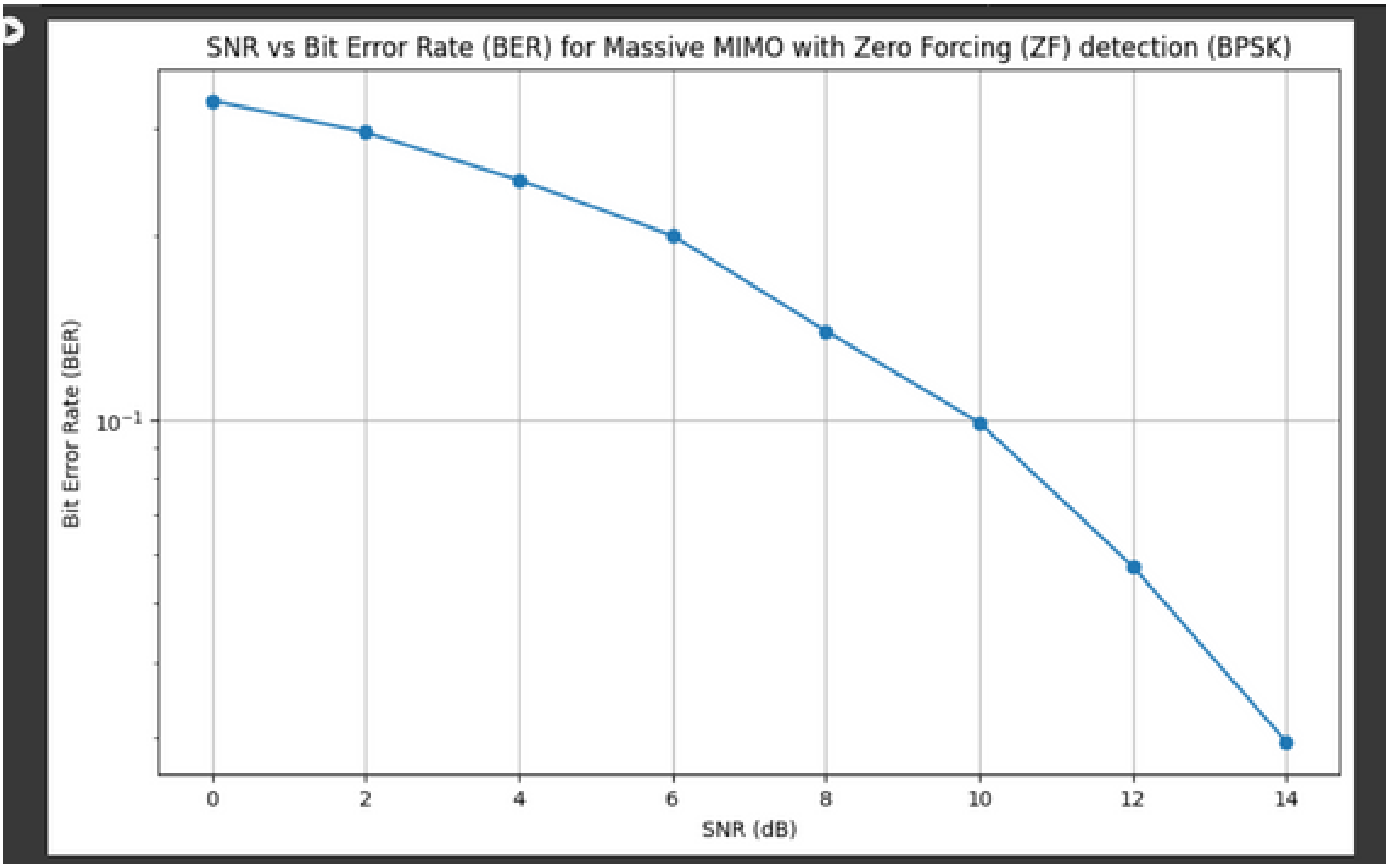
```
import numpy as np

import matplotlib.pyplot as plt


# Function to generate additive white Gaussian noise (AWGN)

def awgn_noise(signal, noise_power):
```

```
    noise = np.random.randn(*signal.shape) * np.sqrt(noise_power)
    return noise

# Function to simulate Massive MIMO system with Zero Forcing (ZF) detection
def simulate_mimo_system(num_antennas, num_users, modulation_order, snr_db, num_trials=10000):
    # Generate random channel matrix H
    H = np.random.randn(num_antennas, num_users) + 1j * np.random.randn(num_antennas, num_users)
    H = H / np.sqrt(2)  # Scale every element by 1/sqrt(2)

    # Generate random symbol vector x_true
    num_symbols = num_users
    x_true = np.random.randint(0, 2, num_symbols) * 2 - 1  # BPSK symbols {-1, 1}

    # Generate AWGN noise power from SNR
    snr_lin = 10**(snr_db / 10)
    noise_power = 1 / snr_lin

    # Initialize lists to store Bit Error Rate (BER) for each SNR value
    snr_values_db = np.arange(0, 16, 2)  # SNR range from -10 dB to 15 dB
    ber_values = []

    for snr_db in snr_values_db:
        # Convert SNR from dB to linear scale
        snr_lin = 10**(snr_db / 10)
        noise_power = 1 / snr_lin

        num_errors = 0
```

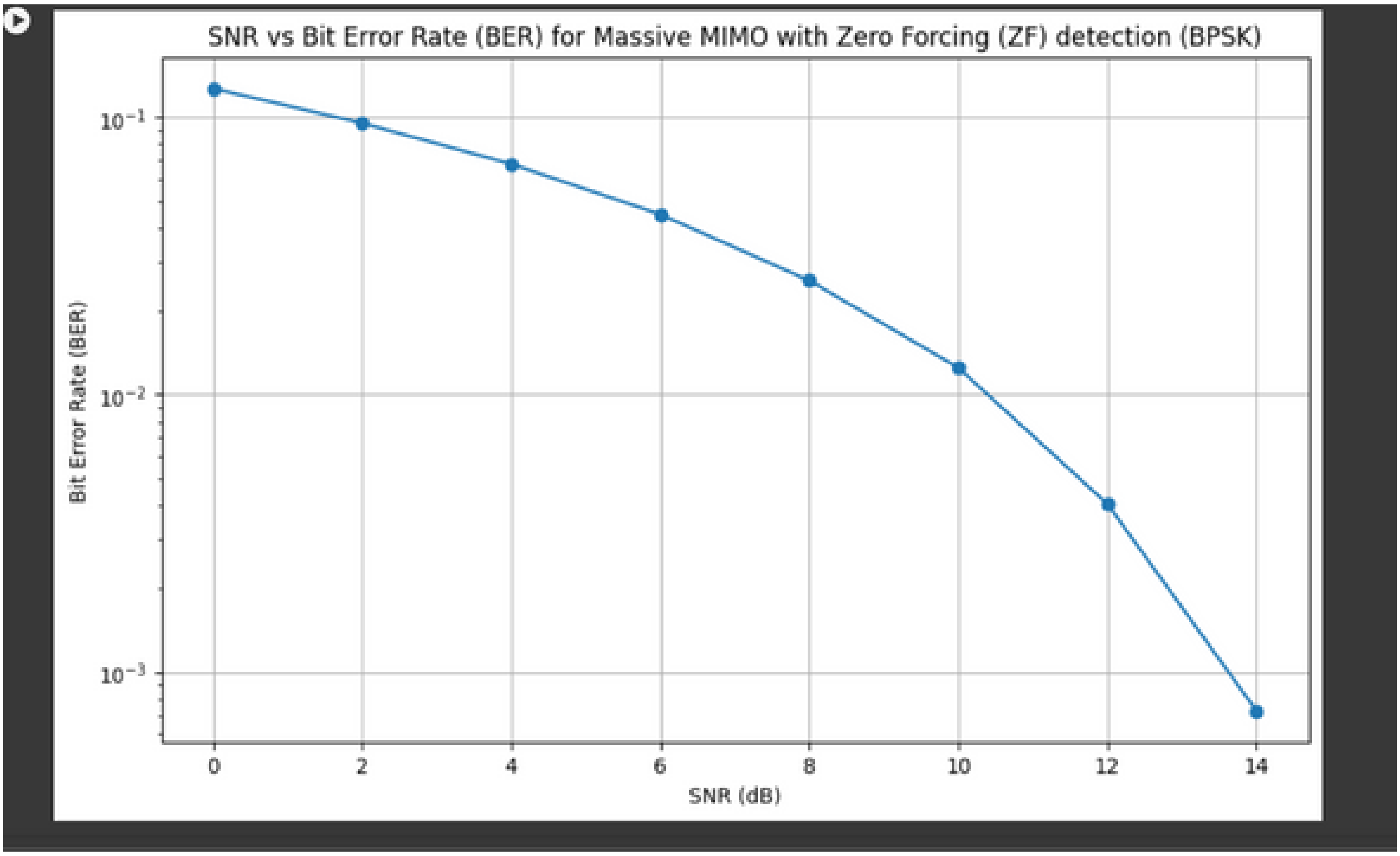
```python
        for _ in range(num_trials):
            # Generate received signal with AWGN
            y = np.dot(H, x_true) + awgn_noise(np.dot(H, x_true), noise_power)

            # Zero Forcing (ZF) detection
            part1_w = np.conj(H.T) @ H
            part2_W = np.linalg.inv(part1_w)
            W_zf = part2_W @ np.conj(H.T)
            x_demod = np.sign(W_zf @ y.real)  # Demodulate symbols

            # Calculate Bit Error Rate (BER)
            num_errors += np.sum(x_demod != x_true)

        ber = num_errors / (num_trials * num_symbols)
        ber_values.append(ber)

    return snr_values_db, ber_values

# Parameters
num_antennas = 64
num_users = 32  # Change this to the desired number of users
modulation_order = 2  # BPSK modulation
num_trials = 10000
snr_db = 10 # Initial SNR value in dB

# Simulate Massive MIMO system with Zero Forcing (ZF) detection for BPSK
snr_values_db, ber_values = simulate_mimo_system(num_antennas, num_users, modulation_order, snr_db, num_trials)
```

```
# Plot SNR vs BER
plt.figure(figsize=(10, 6))
plt.semilogy(snr_values_db, ber_values, marker='o', linestyle='-')
plt.title('SNR vs Bit Error Rate (BER) for Massive MIMO with Zero Forcing (ZF) detection (BPSK)')
plt.xlabel('SNR (dB)')
plt.ylabel('Bit Error Rate (BER)')
plt.grid(True)
plt.show()
```

2x4

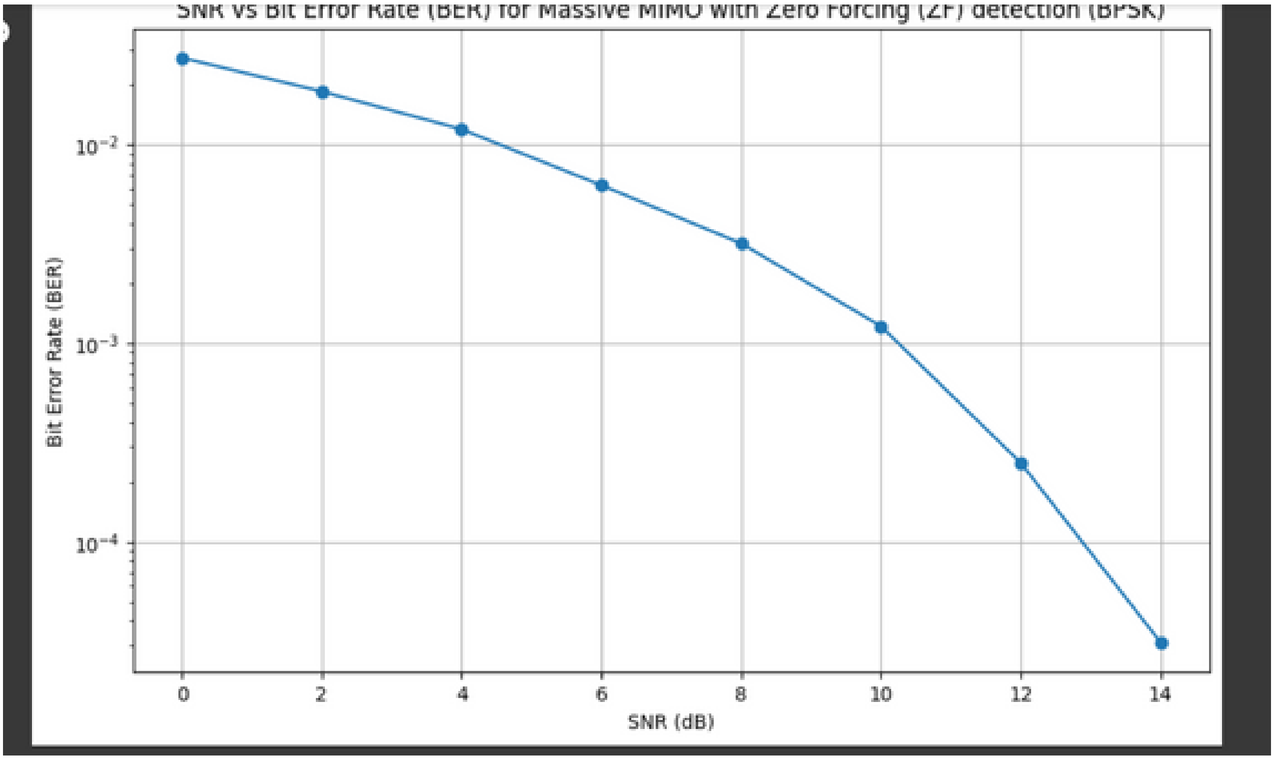


8X 16

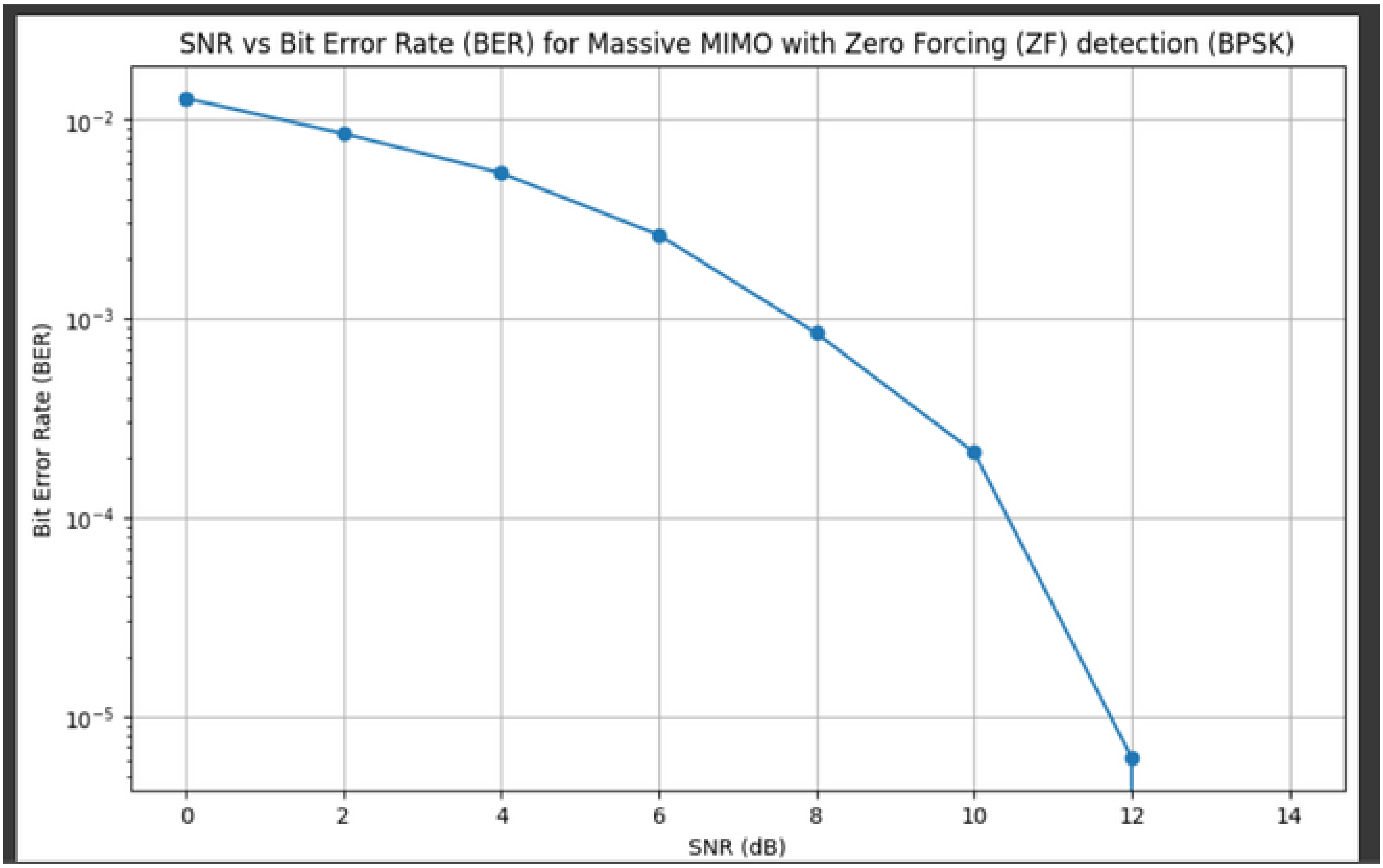


SNR vs Bit Error Rate (BER) for Massive MIMO with Zero Forcing (ZF) detection (BPSK)
Bit Error Rate (BER)
10^{-1}
10^{-2}
10^{-3}
0
2
4
6
8
10
12
14
SNR (dB)

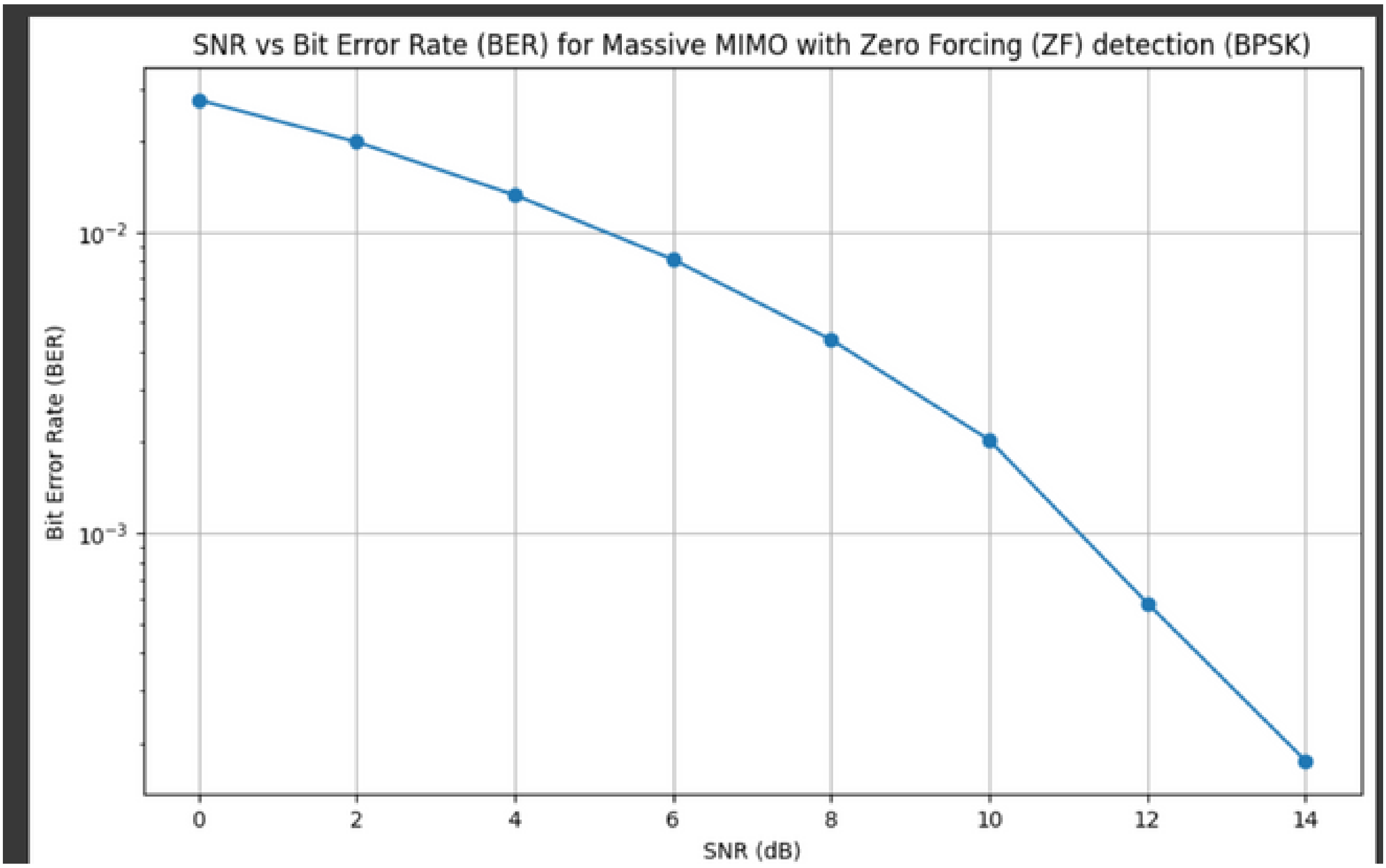
16X32


SNR vs Bit Error Rate (BER) for Massive MIMO with Zero Forcing (ZF) detection (BPSK)
Bit Error Rate (BER)
SNR (dB)

16 x 64


SNR vs Bit Error Rate (BER) for Massive MIMO with Zero Forcing (ZF) detection (BPSK)
Bit Error Rate (BER)
SNR (dB)

32X64



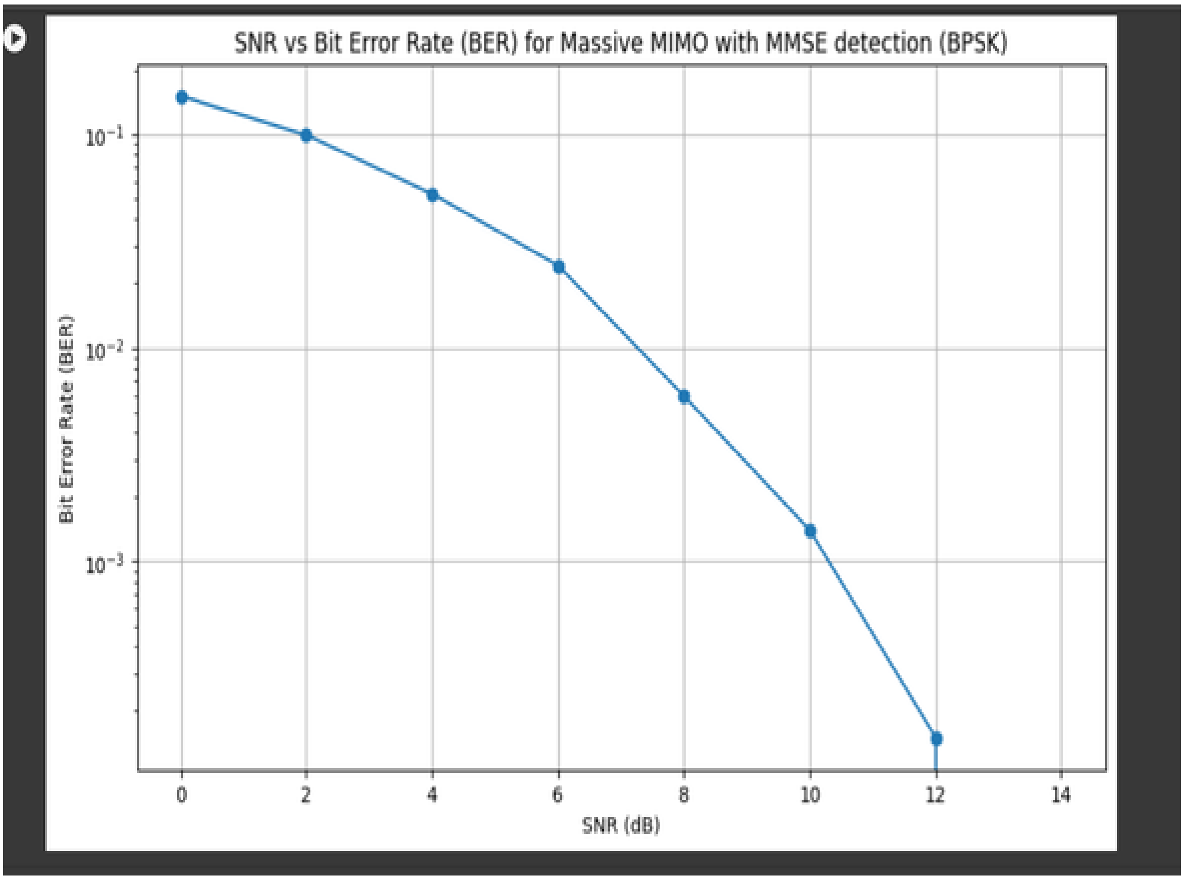
MMSE

```
import numpy as np

import matplotlib.pyplot as plt


# Function to generate additive white Gaussian noise (AWGN)

def awgn_noise(signal, noise_power):

    noise = np.random.randn(*signal.shape) * np.sqrt(noise_power)

    return noise


# Function to simulate Massive MIMO system with MMSE detection

def simulate_mimo_system(num_antennas, num_users, modulation_order, snr_db, num_trials=10000):
```

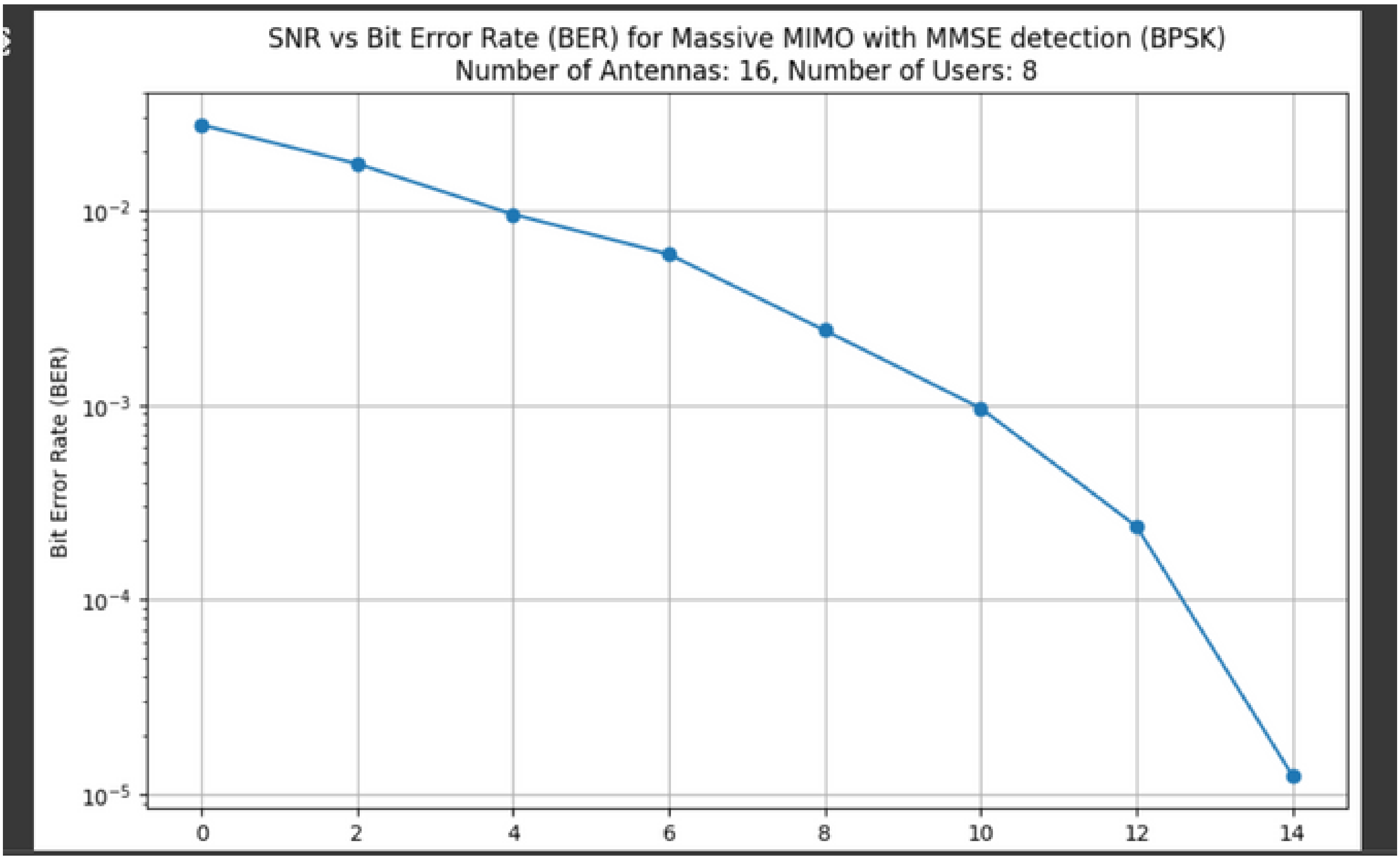
```python
# Generate random channel matrix H
H = np.random.randn(num_antennas, num_users) + 1j * np.random.randn(num_antennas, num_users)
H = H / np.sqrt(2)  # Scale every element by 1/sqrt(2)

# Generate random symbol vector x_true
num_symbols = num_users
x_true = np.random.randint(0, 2, num_symbols) * 2 - 1  # BPSK symbols {-1, 1}

# Generate AWGN noise power from SNR
snr_lin = 10**(snr_db / 10)
noise_power = 1 / snr_lin

# Initialize lists to store Bit Error Rate (BER) for each SNR value
snr_values_db = np.arange(0, 16, 2)  # SNR range from -10 dB to 15 dB
ber_values = []

for snr_db in snr_values_db:
    # Convert SNR from dB to linear scale
    snr_lin = 10**(snr_db / 10)
    noise_power = 1 / snr_lin

    num_errors = 0

    for _ in range(num_trials):
        # Generate received signal with AWGN
        y = np.dot(H, x_true) + awgn_noise(np.dot(H, x_true), noise_power)

        # MMSE detection
```

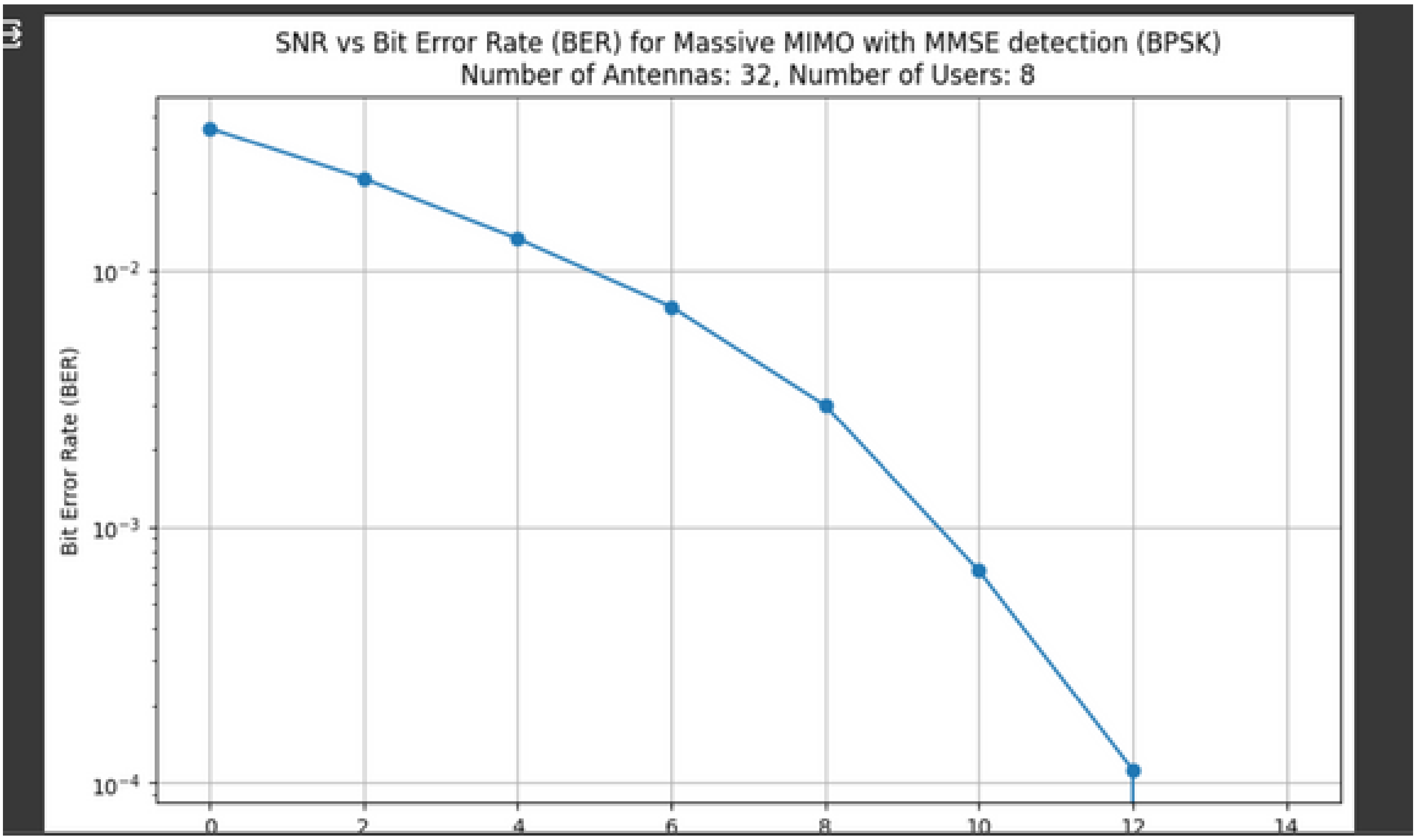
```
            part1_w = np.conj(H.T) @ H

            part2_W = np.linalg.inv(part1_w + noise_power * np.eye(num_users))

            W_mmse = part2_W @ np.conj(H.T)

            x_demod = np.sign(W_mmse @ y.real)  # Demodulate symbols


            # Calculate Bit Error Rate (BER)

            num_errors += np.sum(x_demod != x_true)


        ber = num_errors / (num_trials * num_symbols)

        ber_values.append(ber)


    return snr_values_db, ber_values


# Parameters

num_antennas = 4

num_users = 2 # Change this to the desired number of users

modulation_order = 2  # BPSK modulation

num_trials = 10000

snr_db = 10  # Initial SNR value in dB


# Simulate Massive MIMO system with MMSE detection for BPSK

snr_values_db, ber_values = simulate_mimo_system(num_antennas, num_users, modulation_order, snr_db,
num_trials)


# Plot SNR vs BER

plt.figure(figsize=(10, 6))

plt.semilogy(snr_values_db, ber_values, marker='o', linestyle='-')

plt.title('SNR vs Bit Error Rate (BER) for Massive MIMO with MMSE detection (BPSK)')

plt.xlabel('SNR (dB)')
```

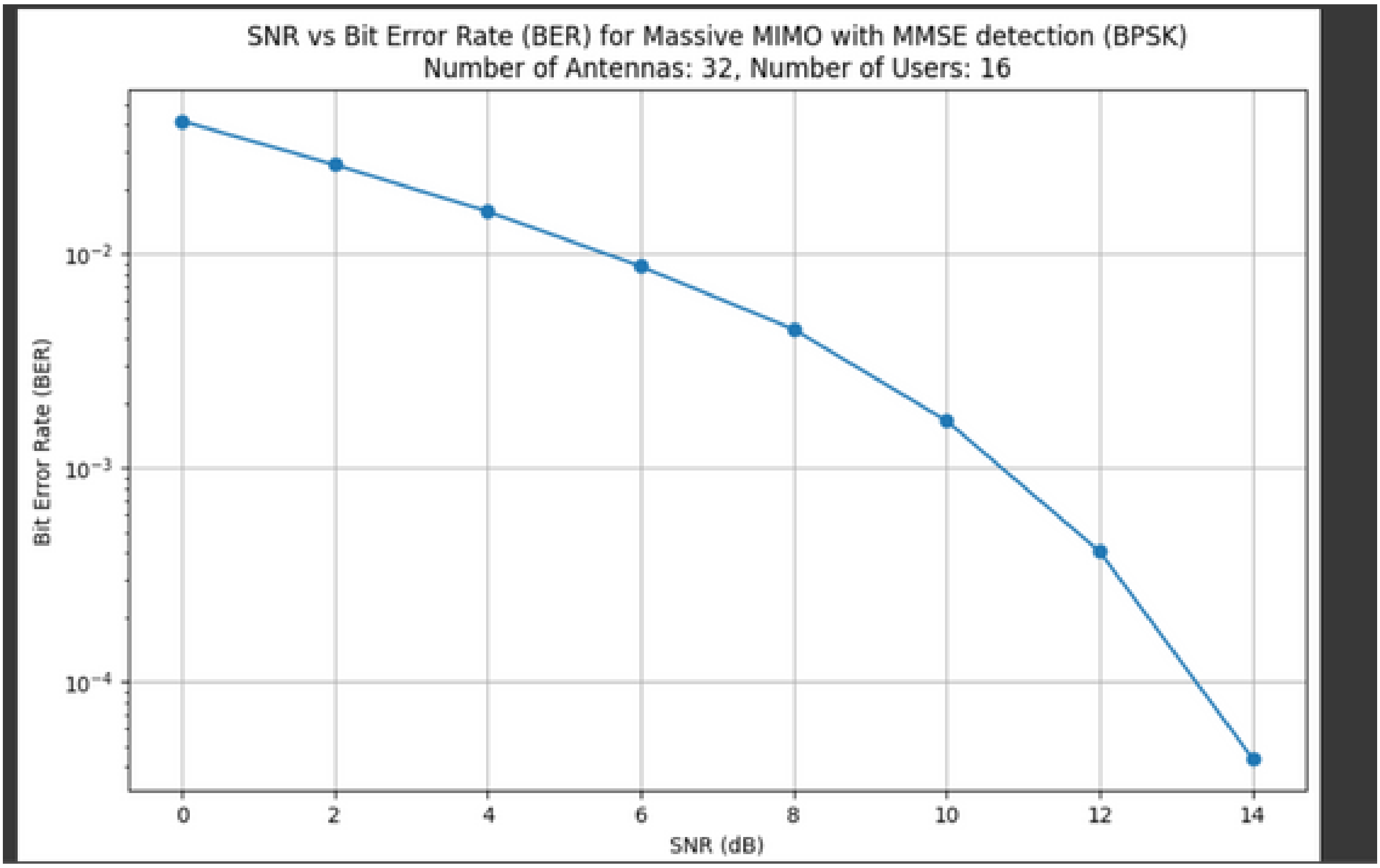
```
plt.ylabel('Bit Error Rate (BER)')
plt.grid(True)
plt.show()
```

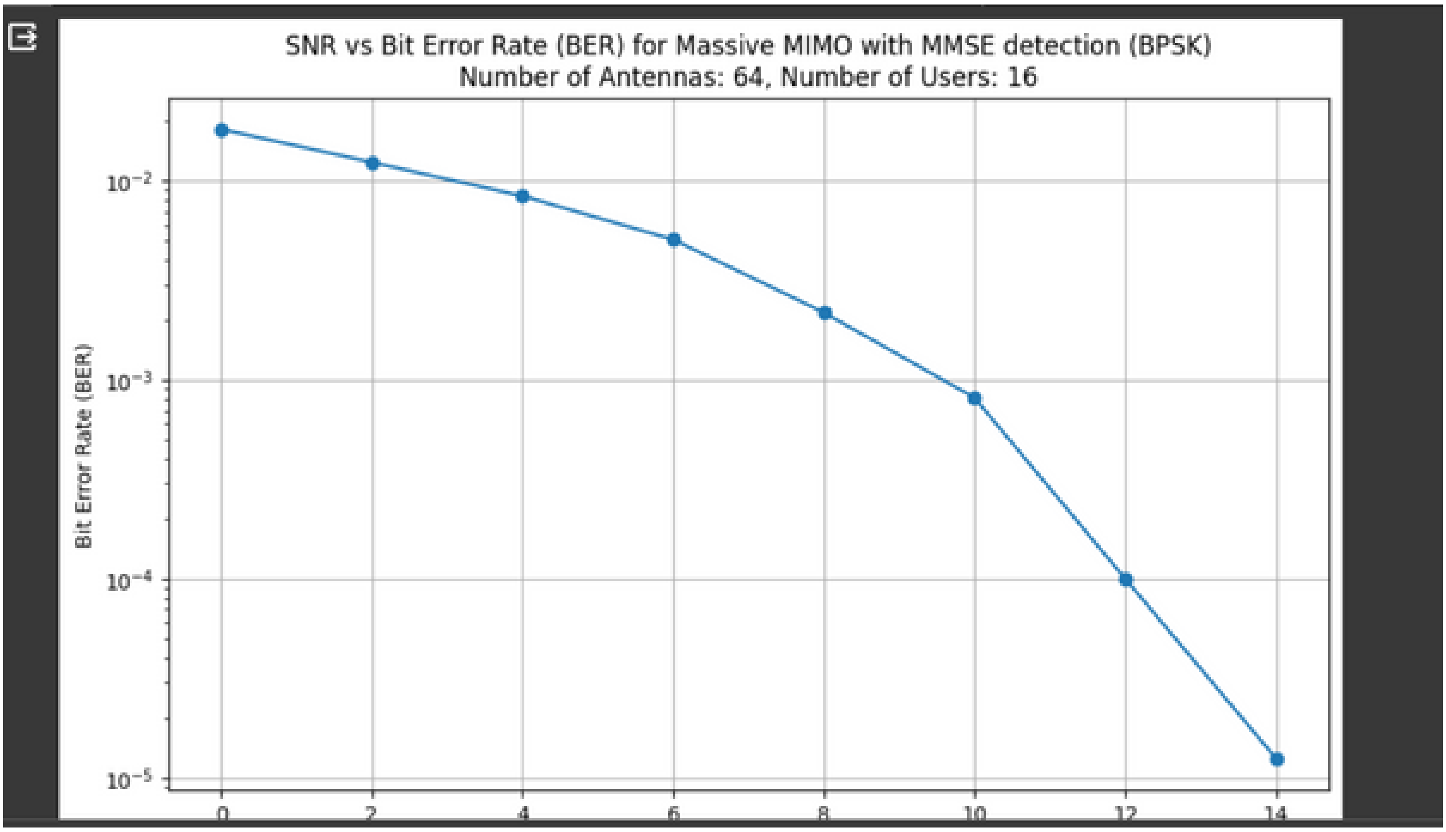


2X4


SNR vs Bit Error Rate (BER) for Massive MIMO with MMSE detection (BPSK)
Number of Antennas: 16, Number of Users: 8
Bit Error Rate (BER)
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
2
4
6
8
10
12
14


SNR vs Bit Error Rate (BER) for Massive MIMO with MMSE detection (BPSK)
Number of Antennas: 32, Number of Users: 8
Bit Error Rate (BER)
10^{-2}
10^{-3}
10^{-4}
0
2
4
6
8
10
12
14

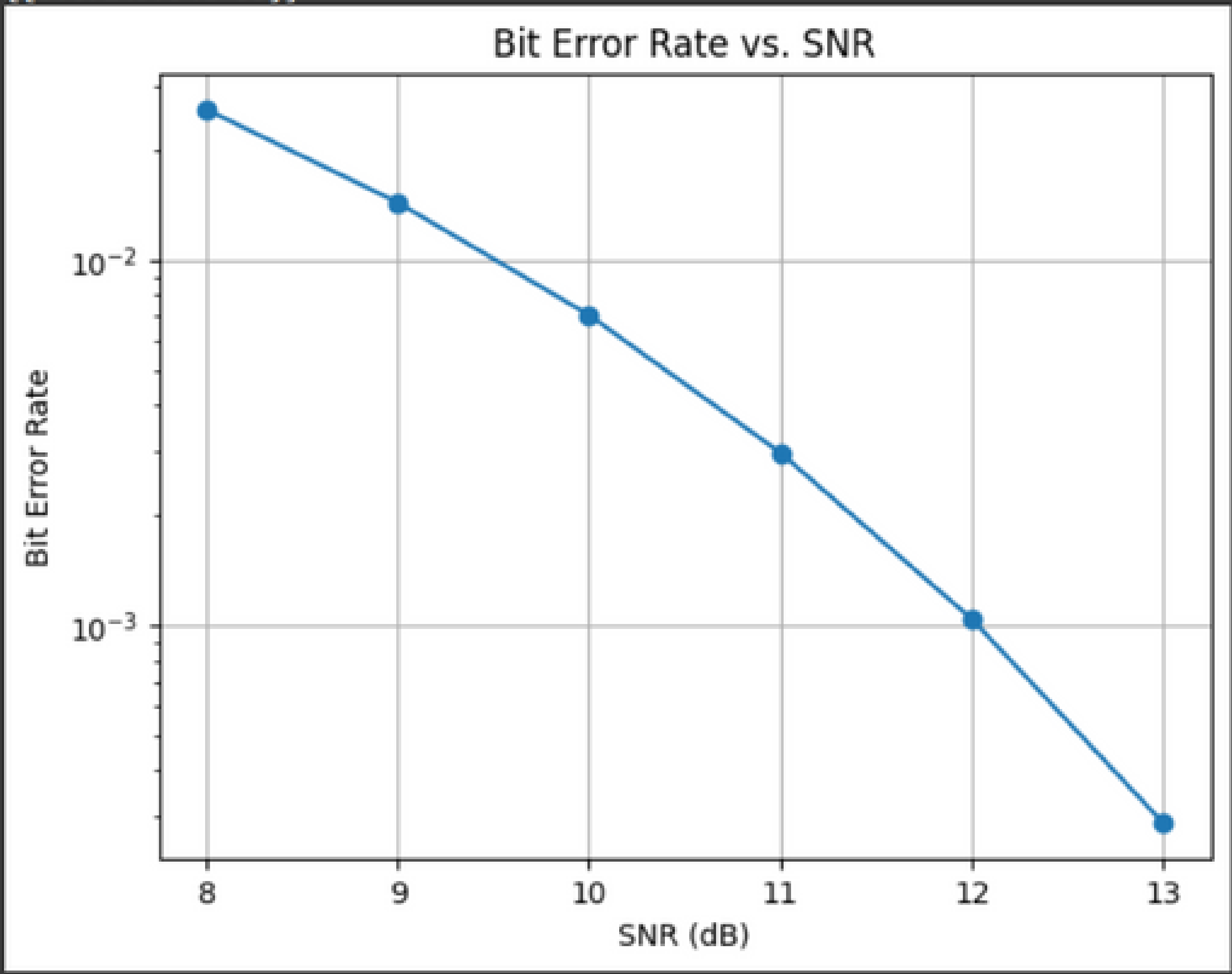

SNR vs Bit Error Rate (BER) for Massive MIMO with MMSE detection (BPSK)
Number of Antennas: 32, Number of Users: 16
Bit Error Rate (BER)
10^{-2}
10^{-3}
10^{-4}
0
2
4
6
8
10
12
14
SNR (dB)



SNR vs Bit Error Rate (BER) for Massive MIMO with MMSE detection (BPSK)
Number of Antennas: 64, Number of Users: 16
Bit Error Rate (BER)
10⁻²
10⁻³
10⁻⁴
10⁻⁵
0
2
4
6
8
10
12
14

Detnet

```
[[3.67249794e-06]]
```



- **COMBINED PLOT FOR ZERO FORCING WITH CONSTANT N (No. of antennas) and varying K no of users**

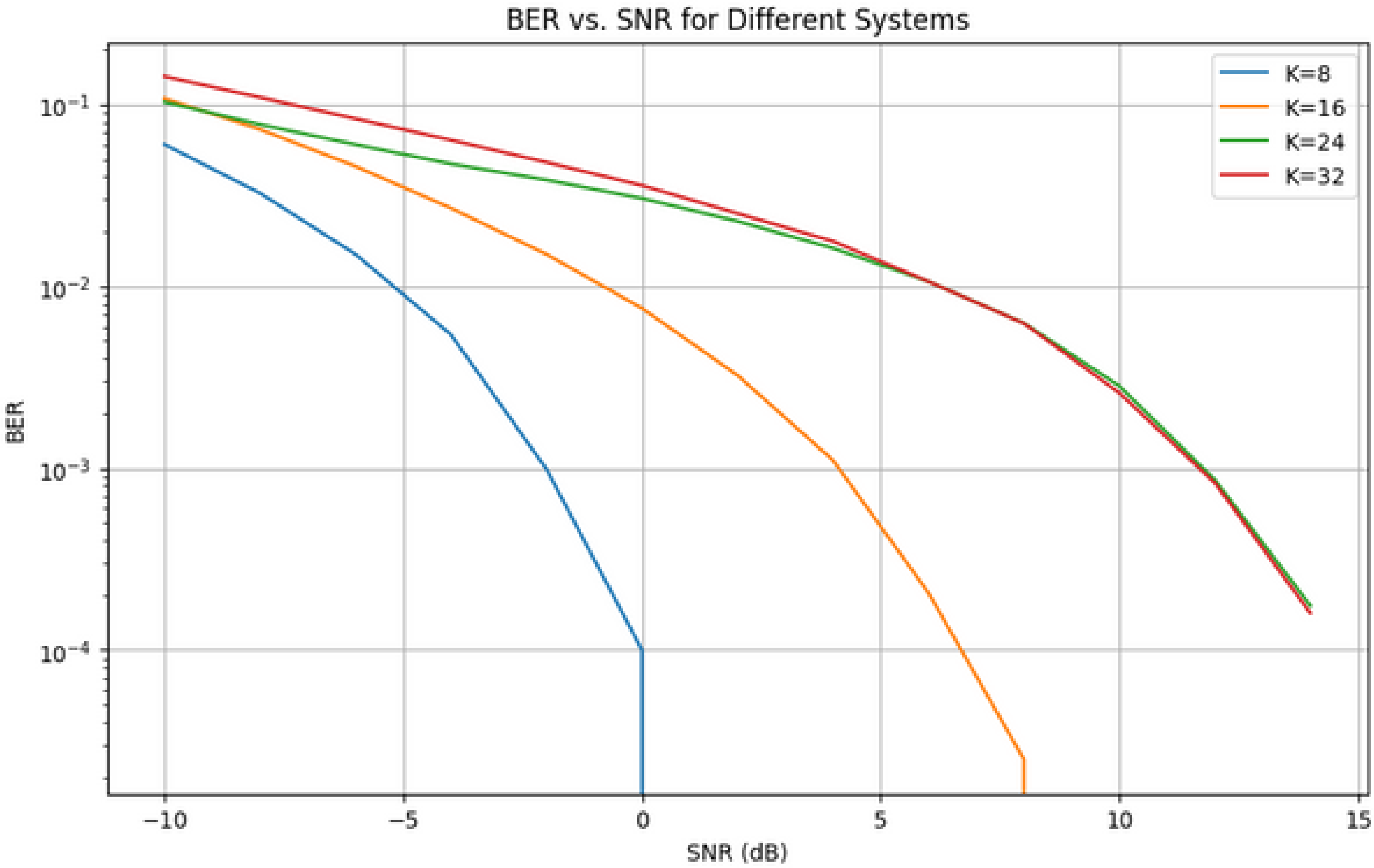


FIG 1) ZERO FOCRING for N=64 and varying K

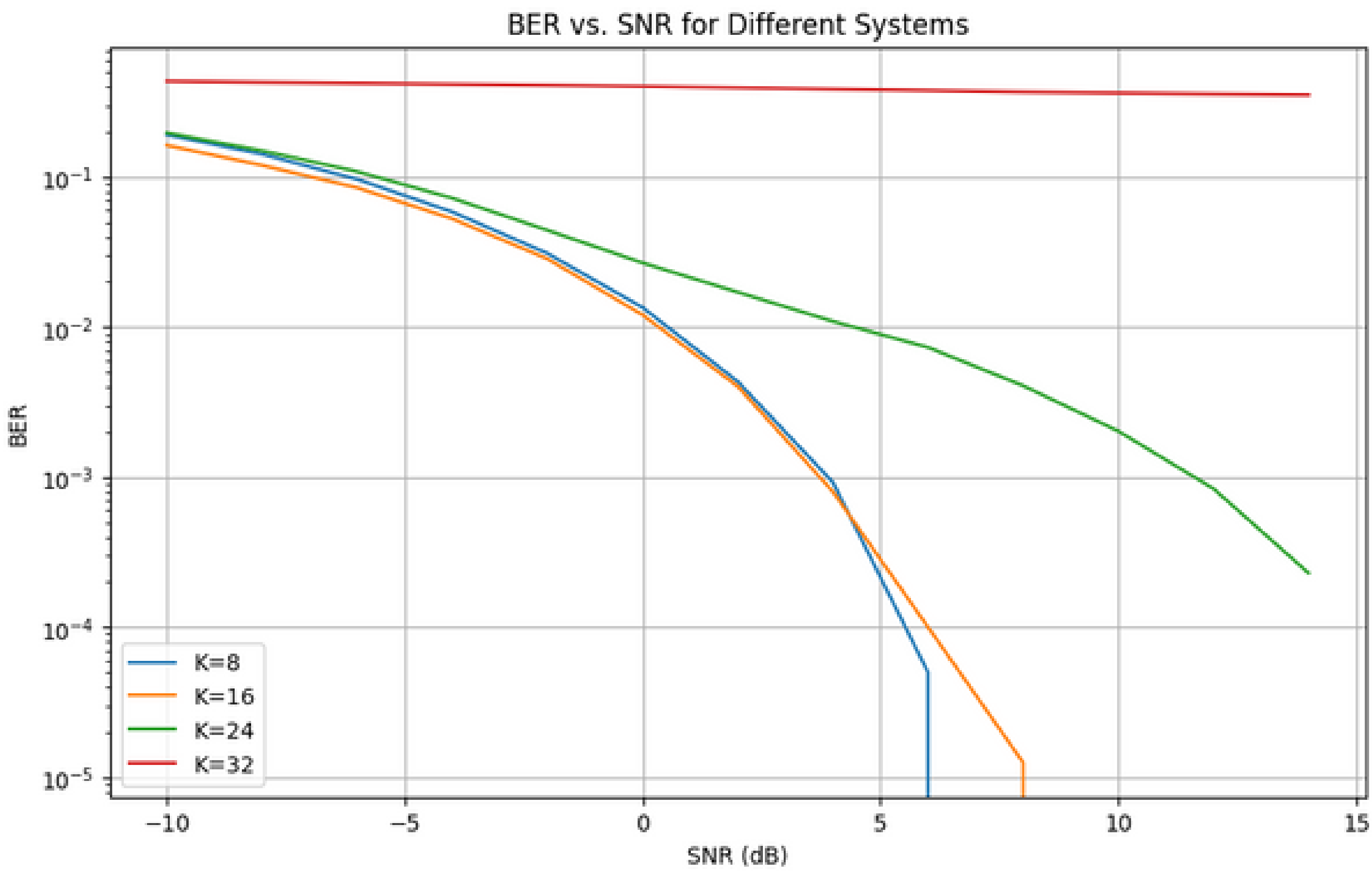
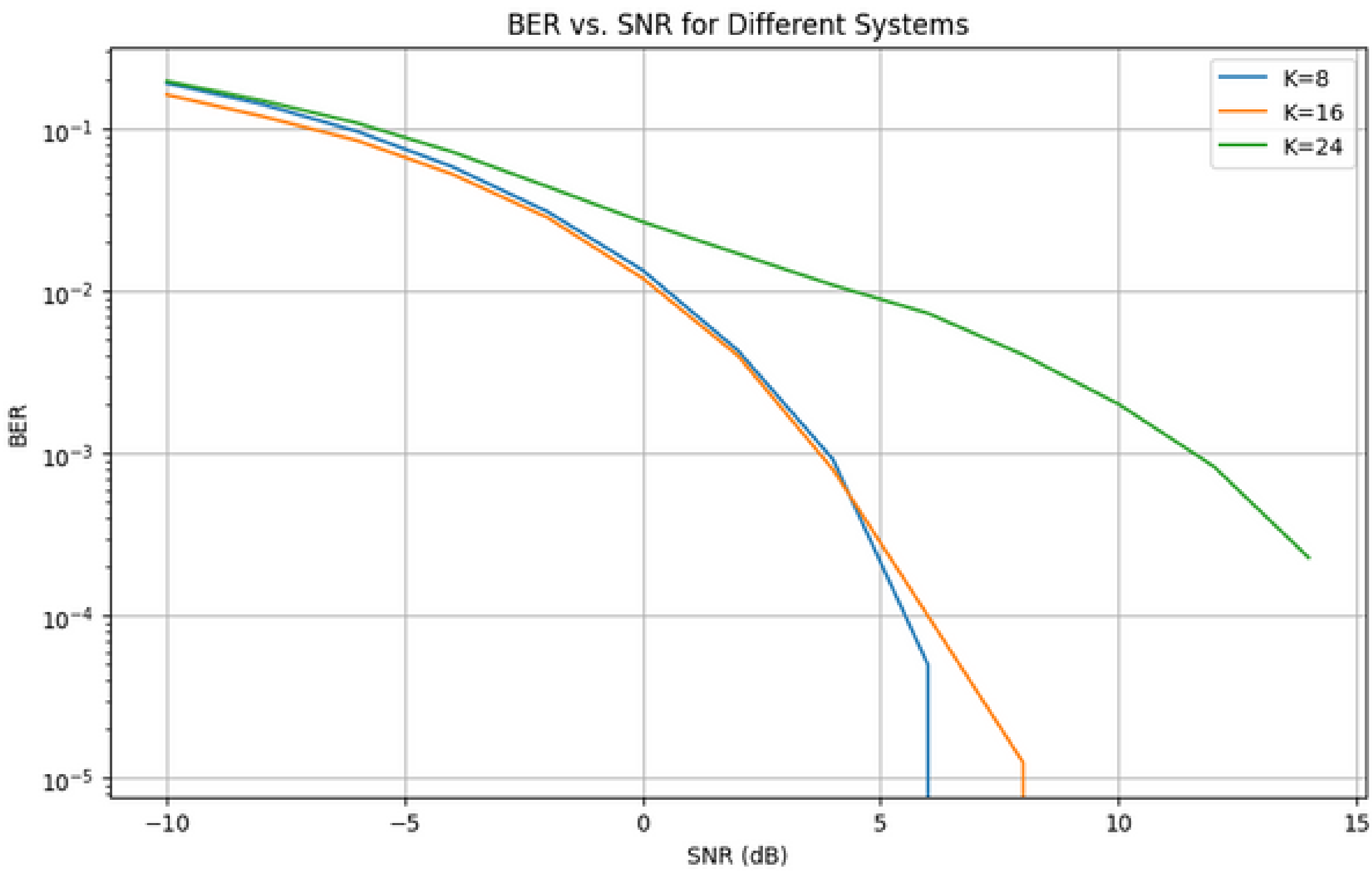
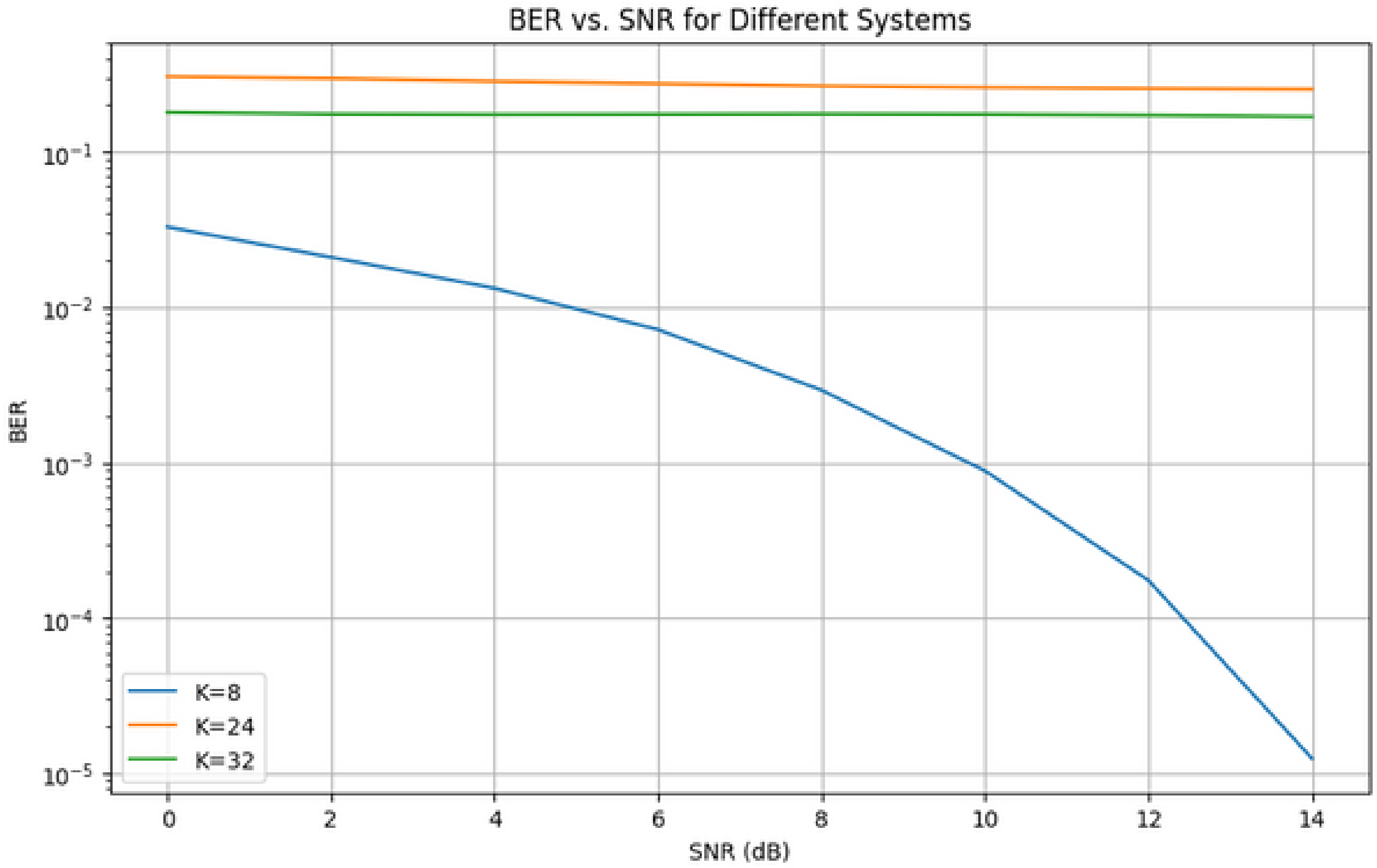


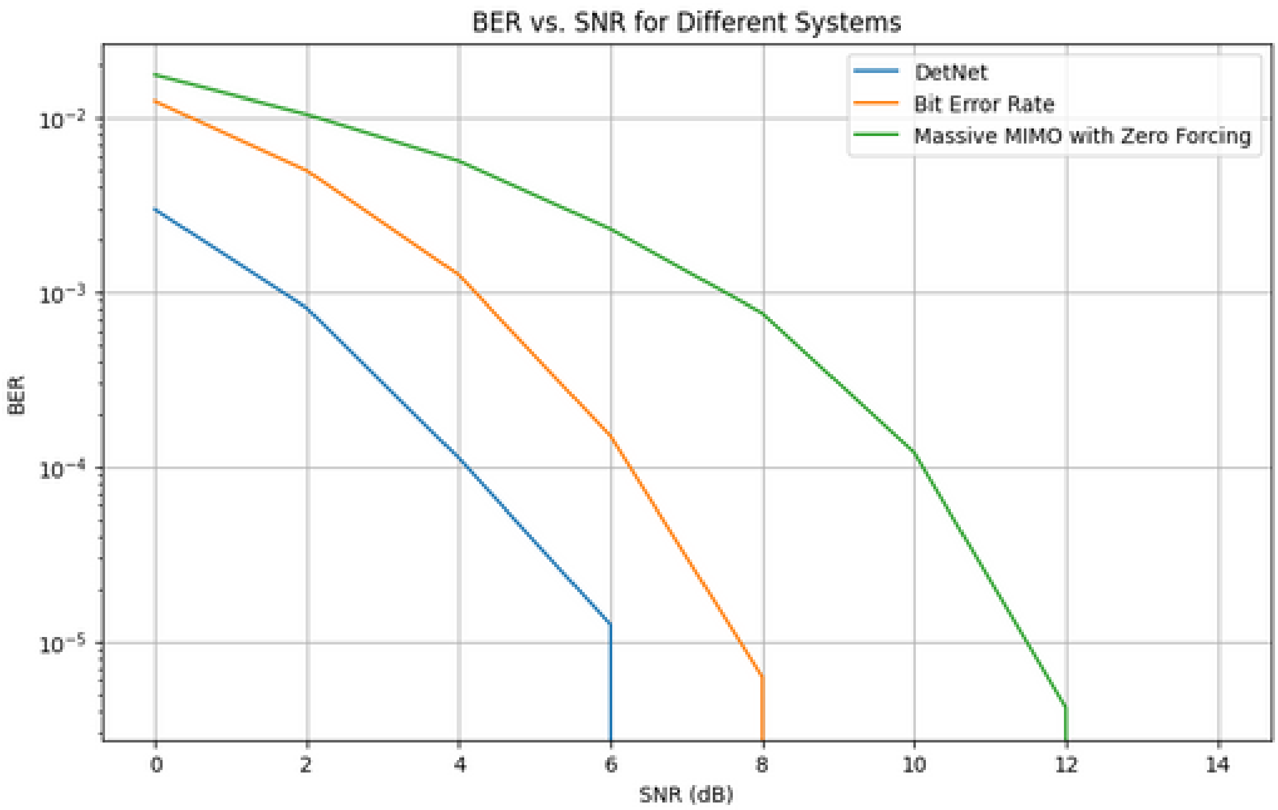
Fig2) ZF for N=32 and varying K(ALL)





[ZF] N=16 N K USERS

•



- **MMSE for fixed N and varying K**

FIG1) MMSE FOR N =64 and varying K

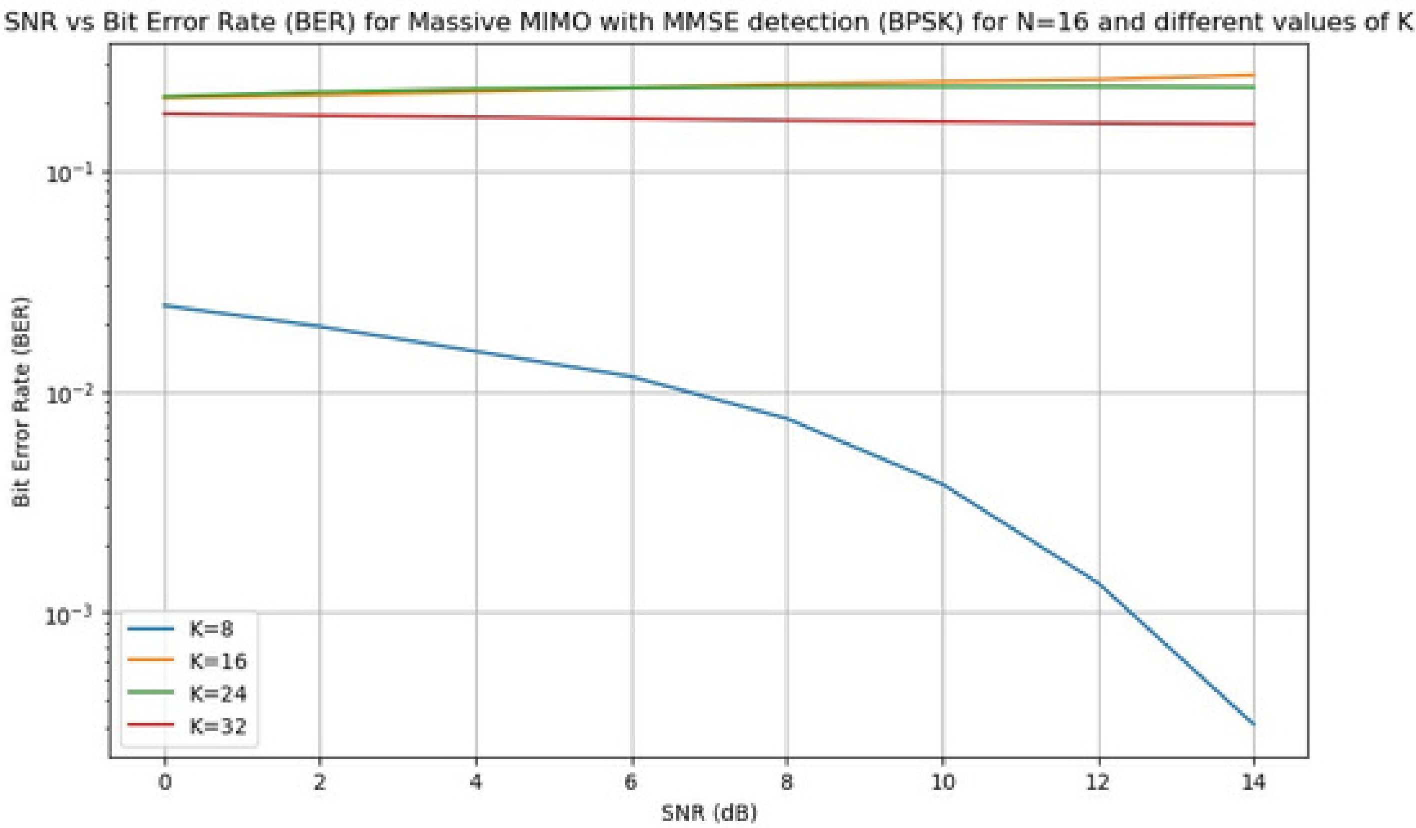


FIG 2) MMSE FOR N=16 AND DIFF K

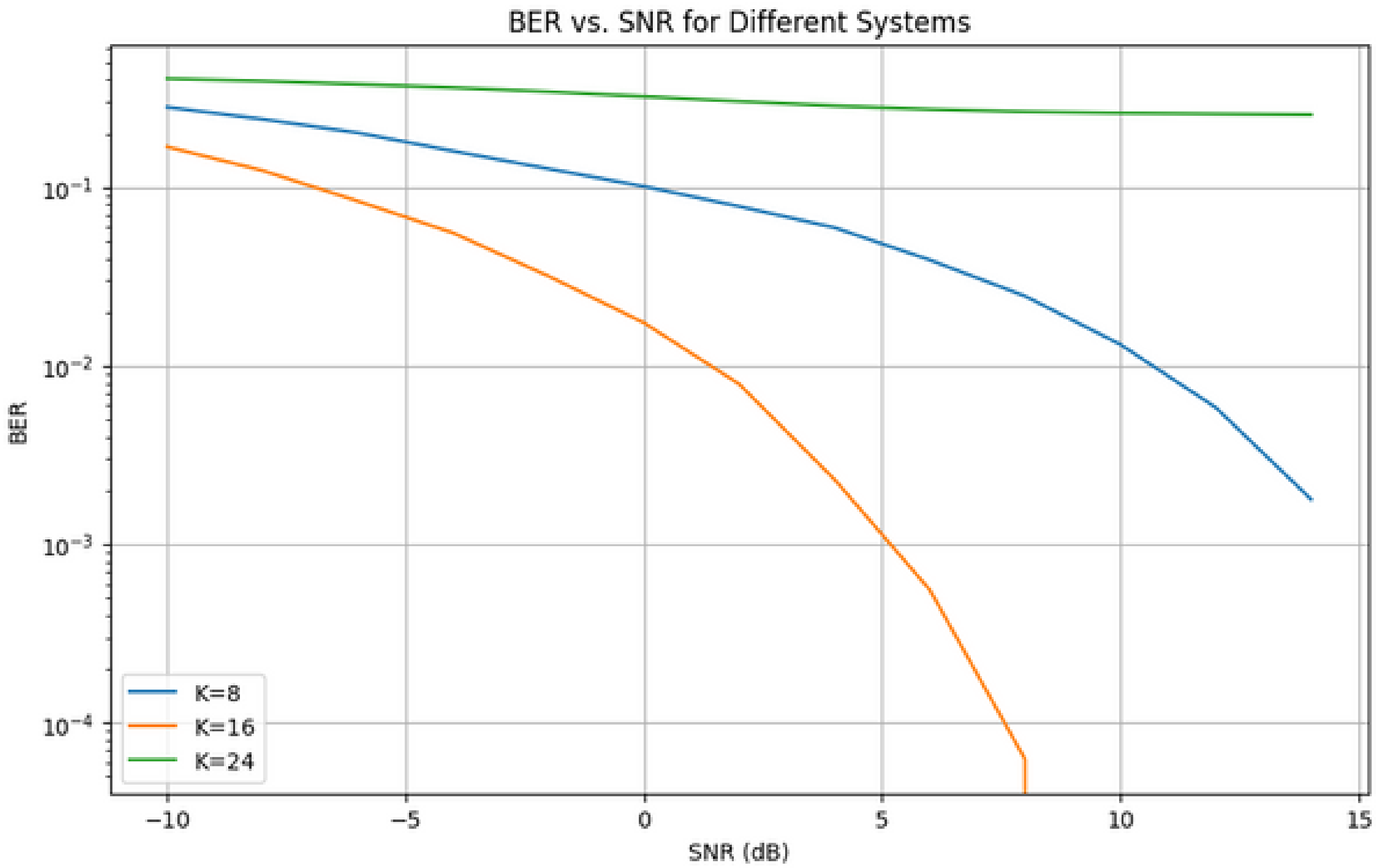
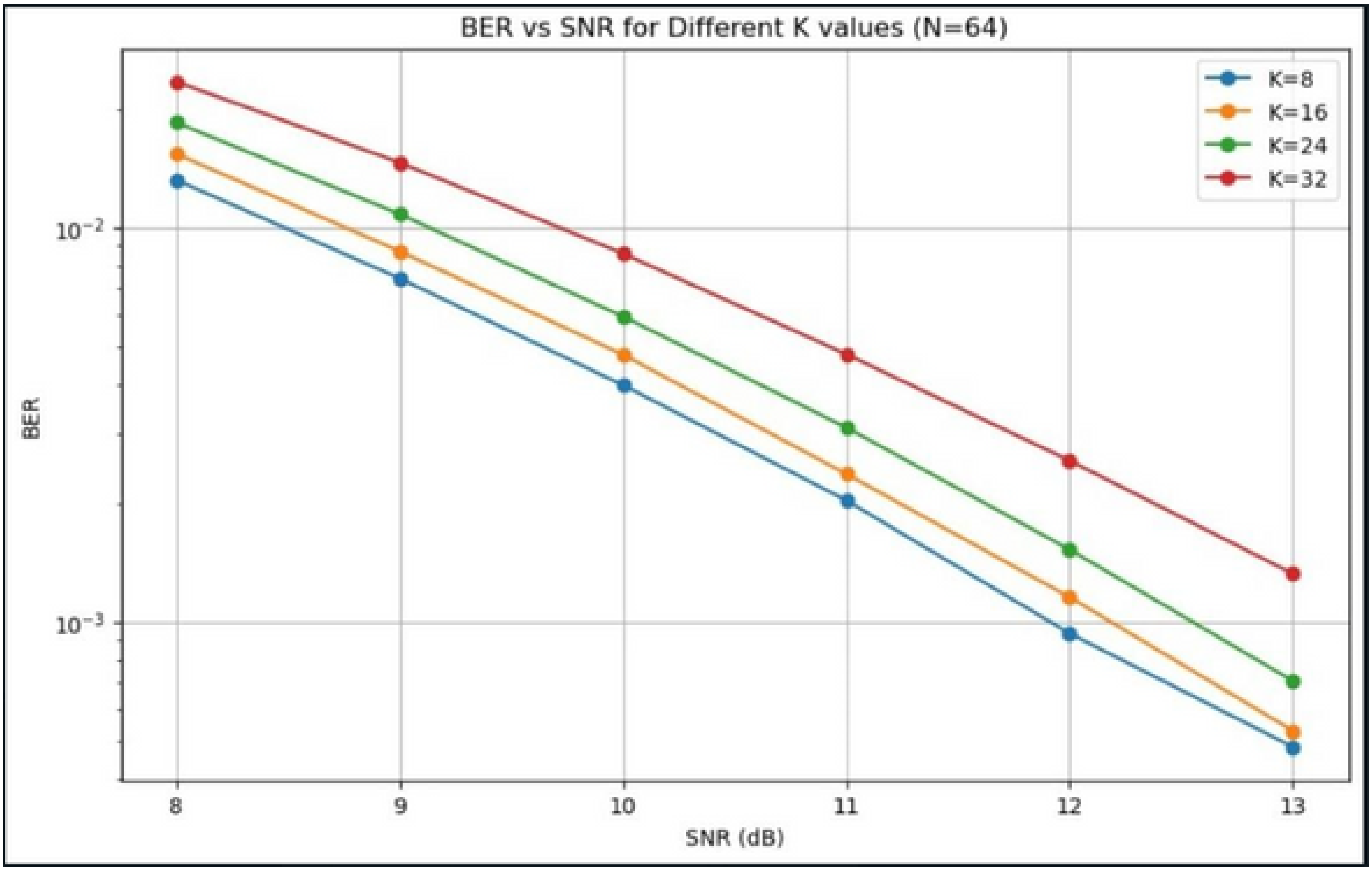
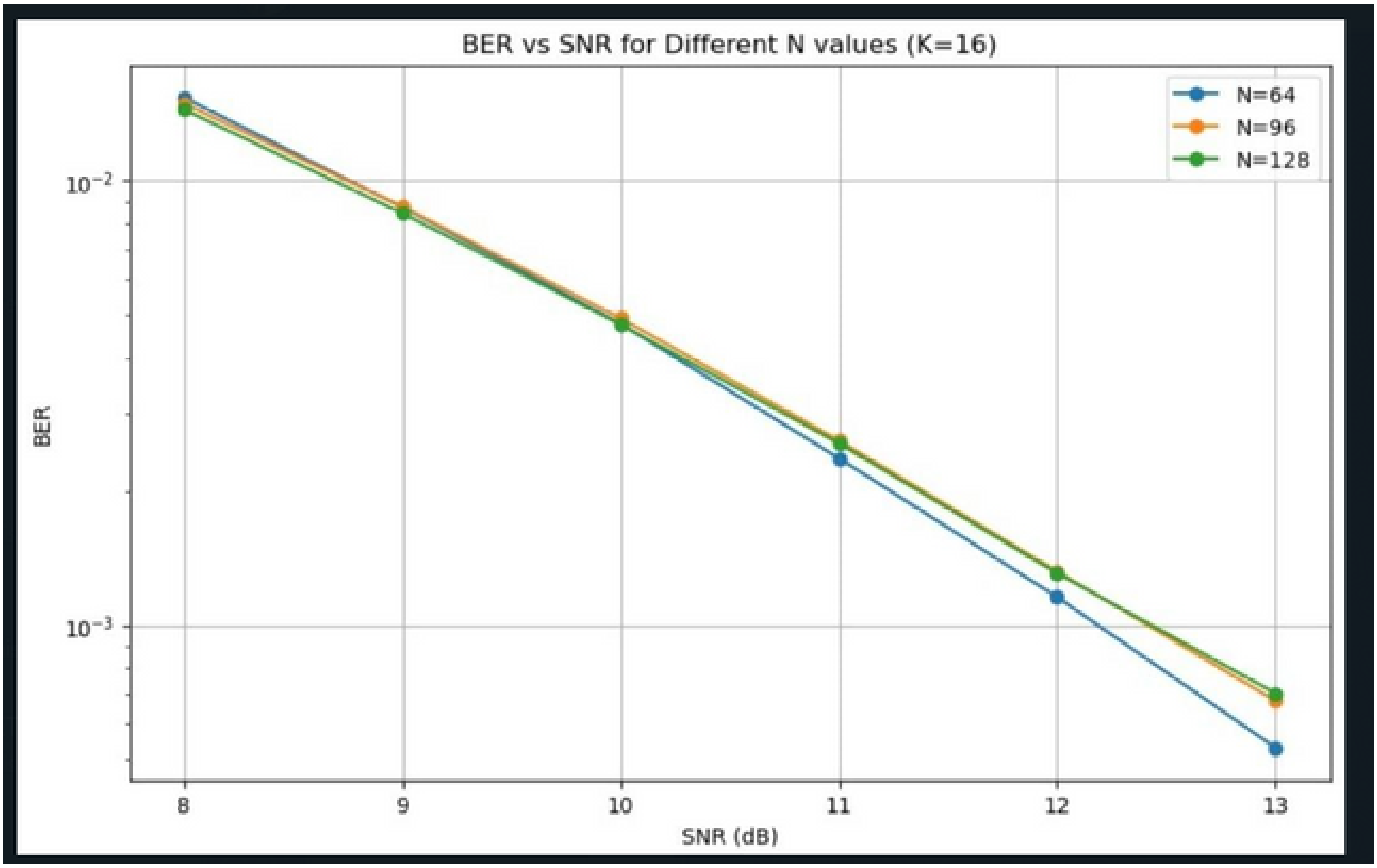


Fig 3) N=32 FOR MMSE

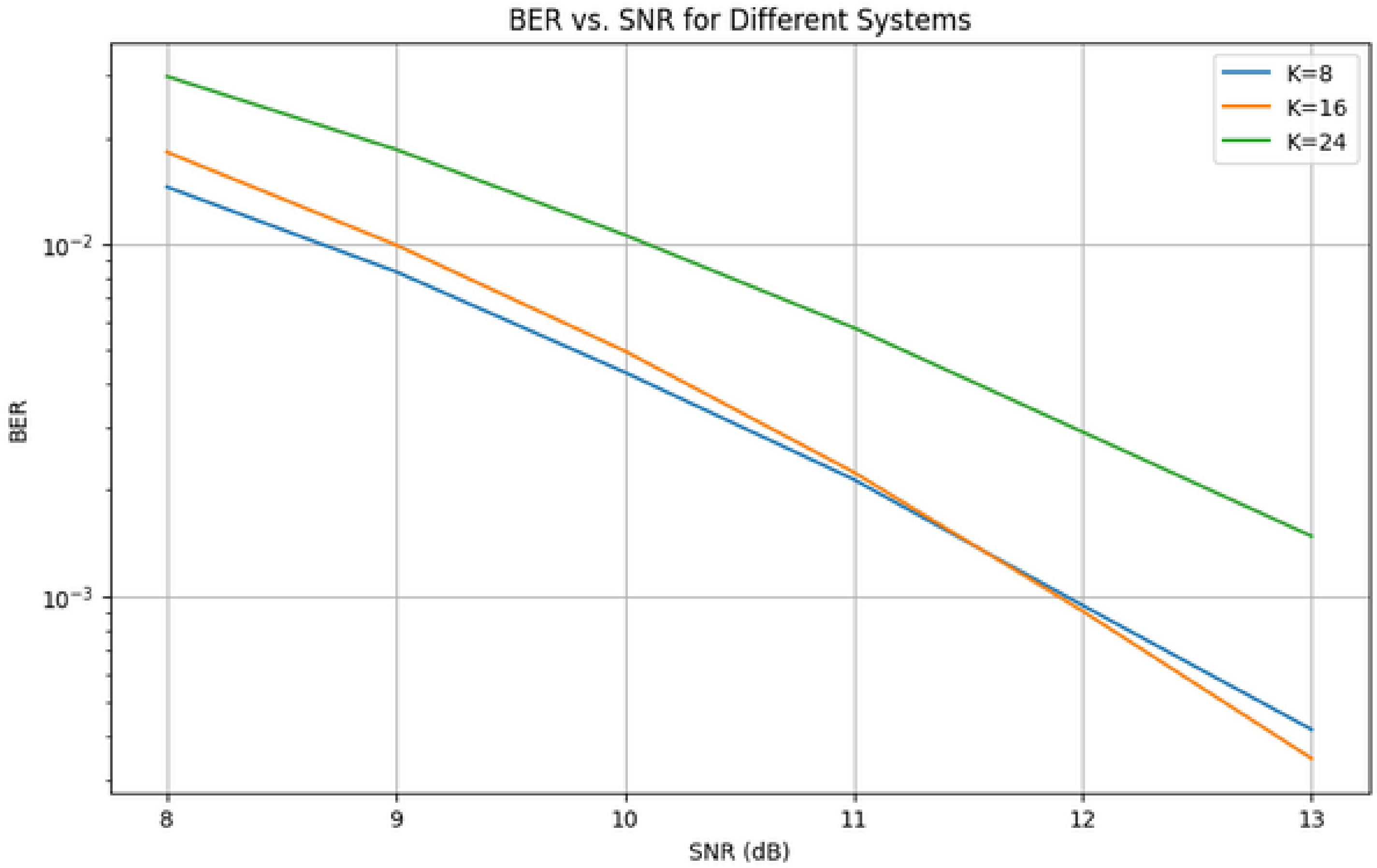


DETNET

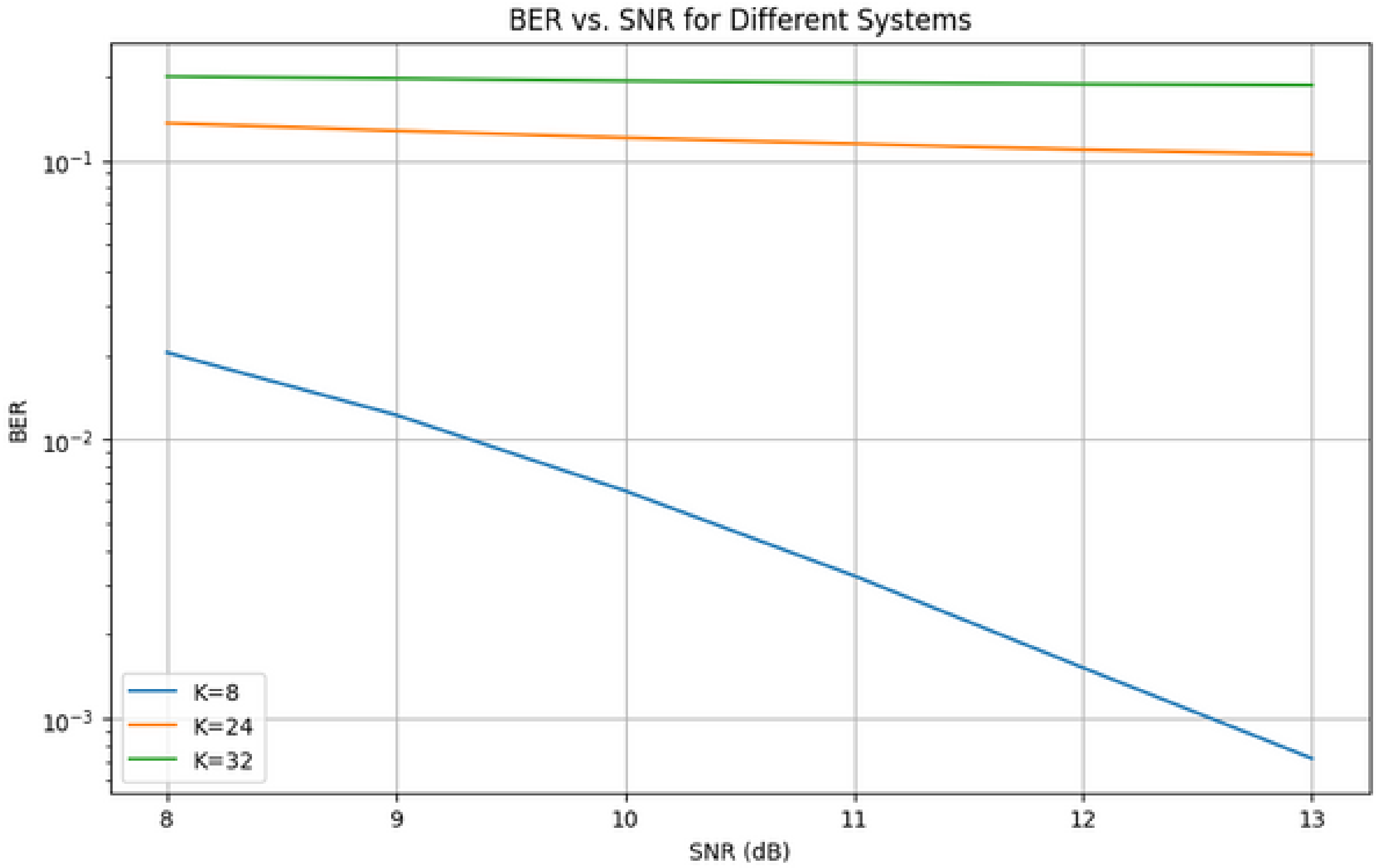
Fig 1) For N =64 and varying K



For K =16 and varying N



DETNET N=32 AND VARYING K



DETNET N=16 AND VARYING K

FIG 3) ZF for N=32 and K=8,16,24,32