

**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

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)')

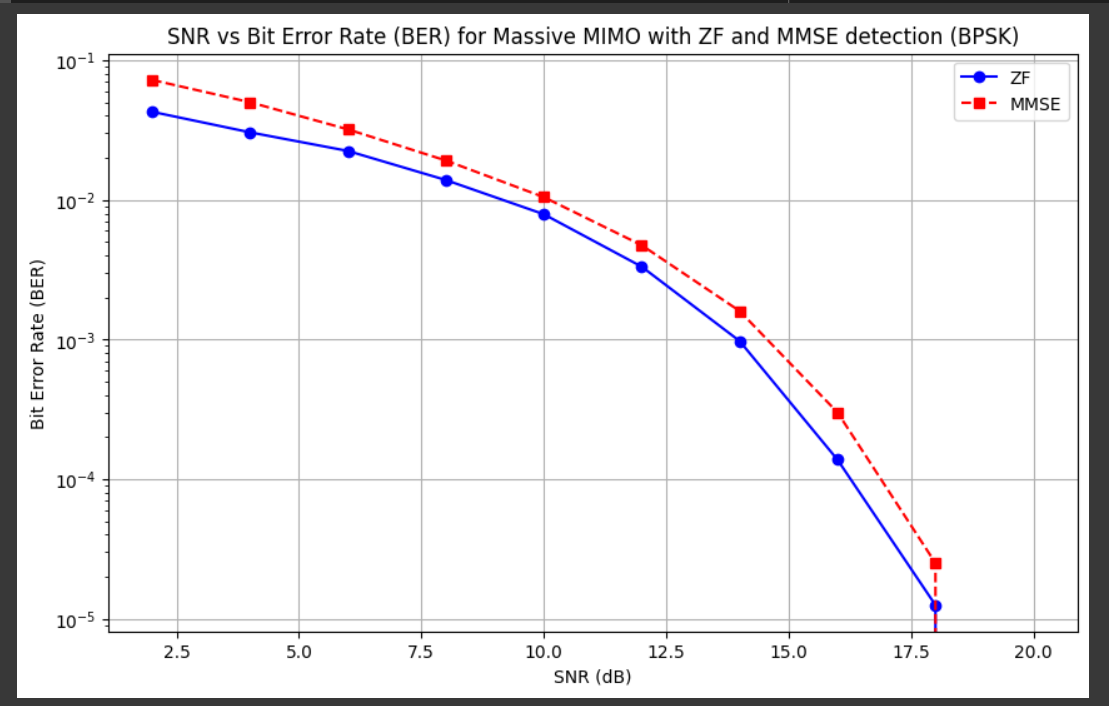
plt.ylabel('Bit Error Rate (BER)')

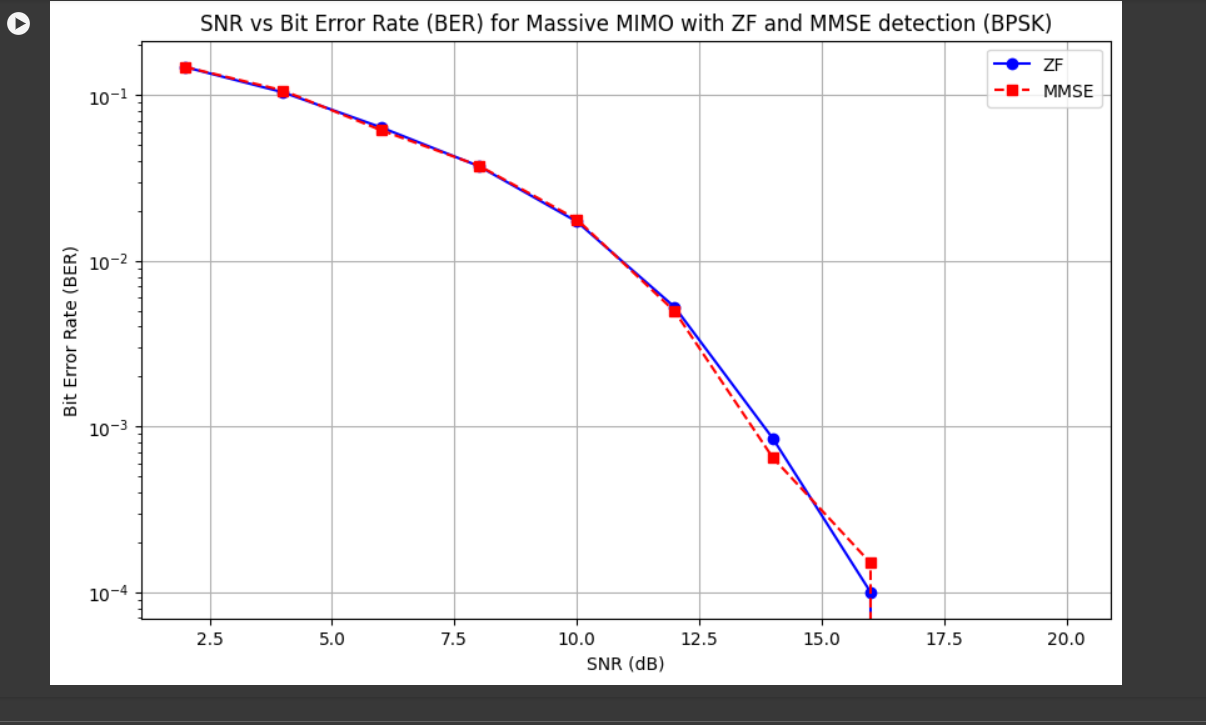
plt.legend()

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

        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)')

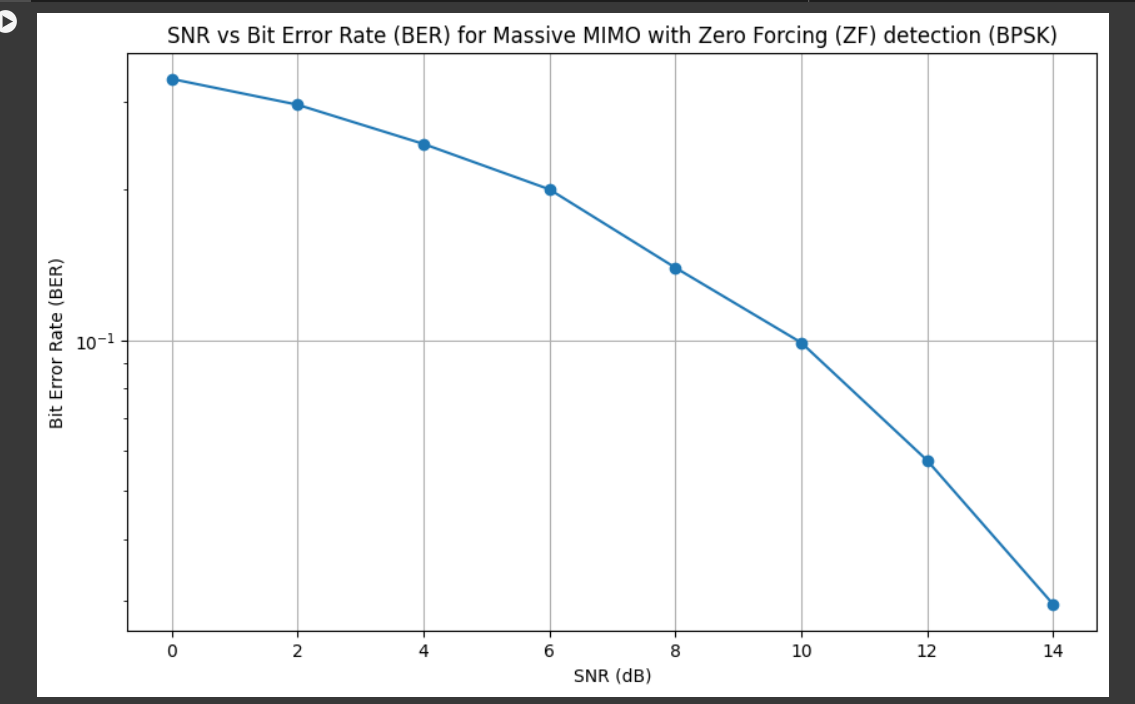
plt.xlabel('SNR (dB)')

plt.ylabel('Bit Error Rate (BER)')

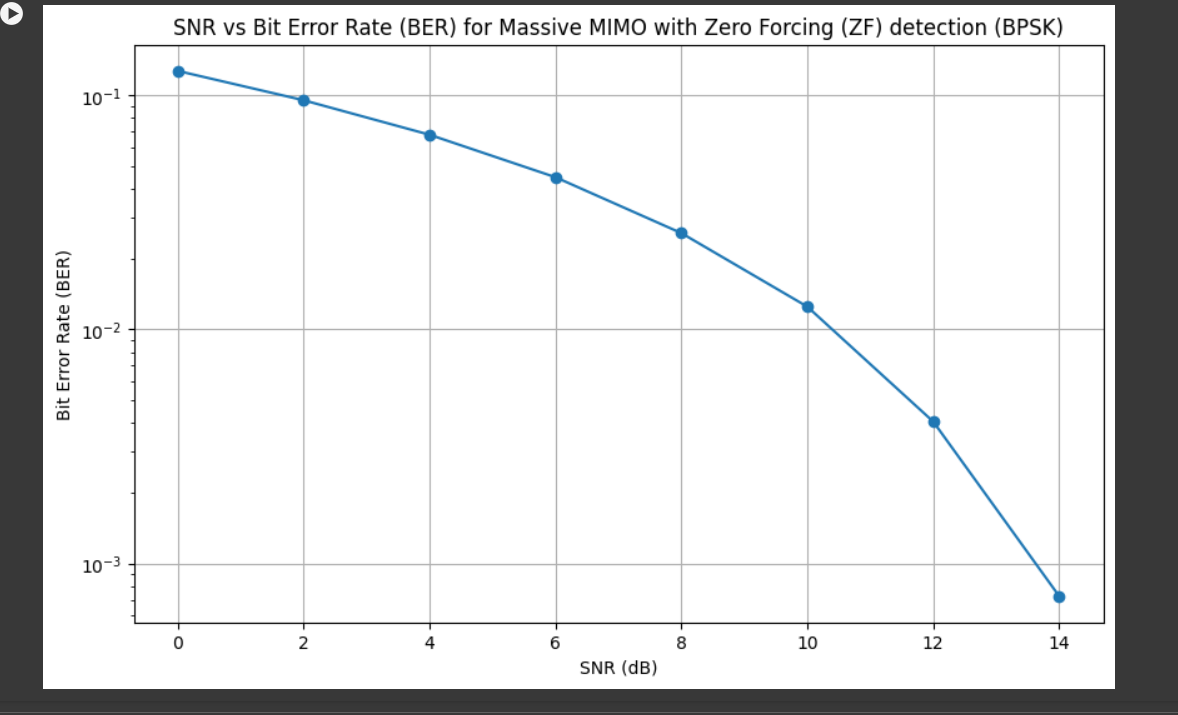
plt.grid(True)

plt.show()

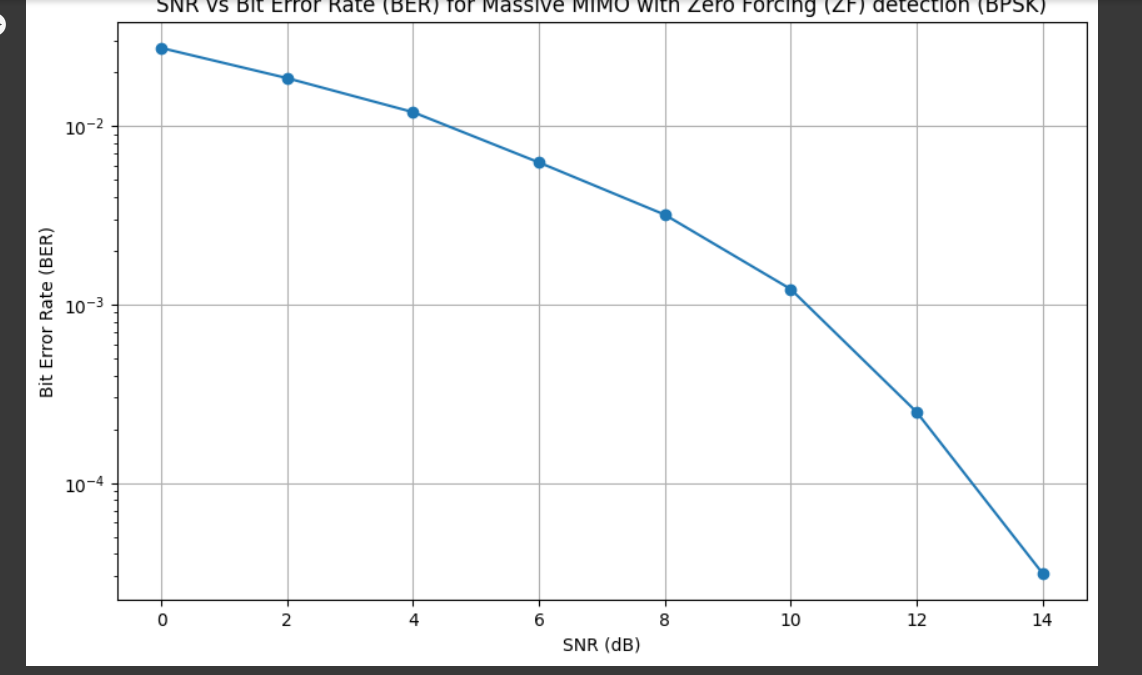
**2x4**



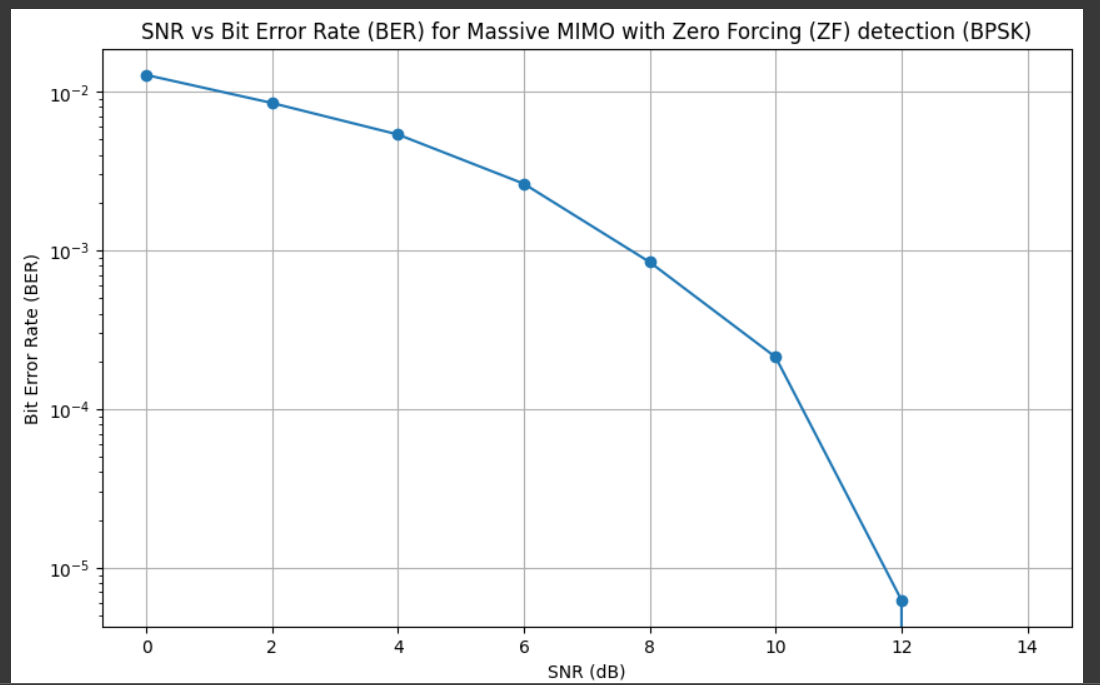
8X 16



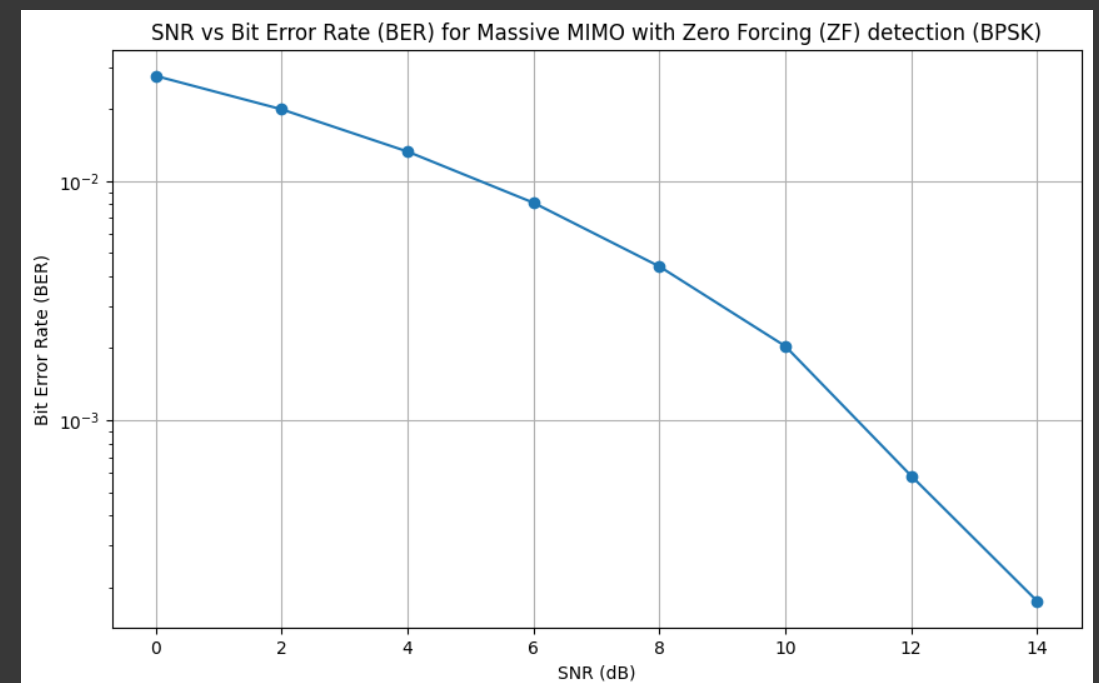
16X32



**16 x 64**

****

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):

    # 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)')

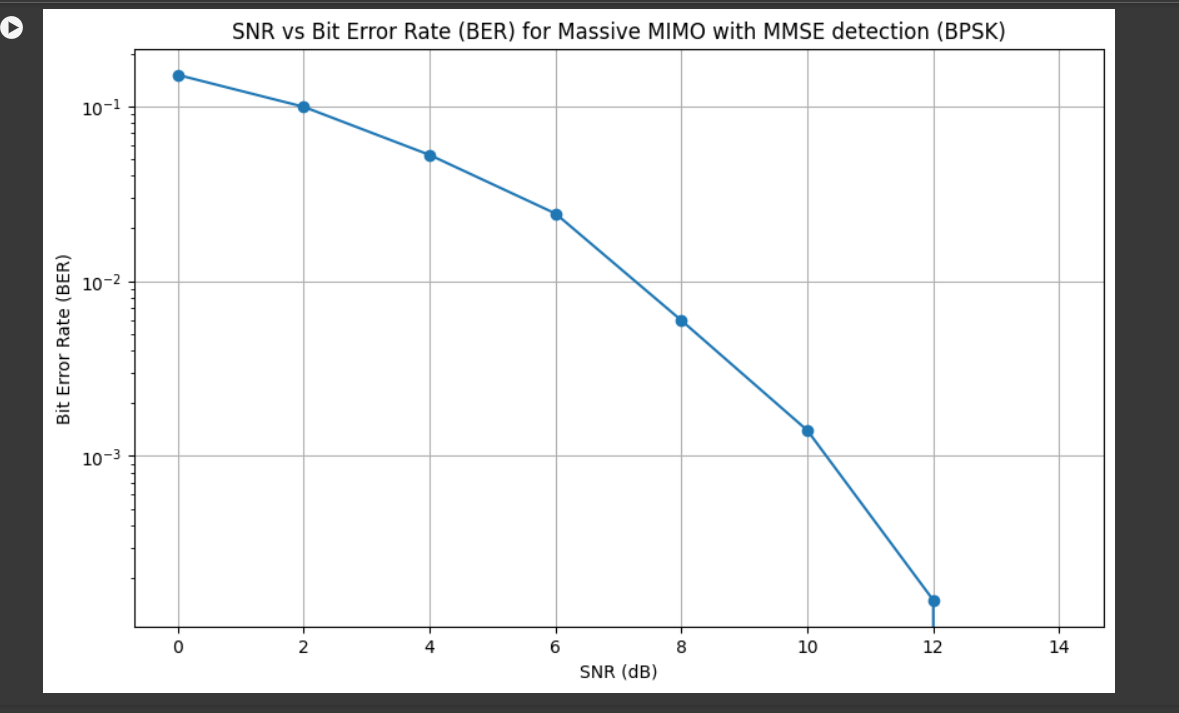
plt.xlabel('SNR (dB)')

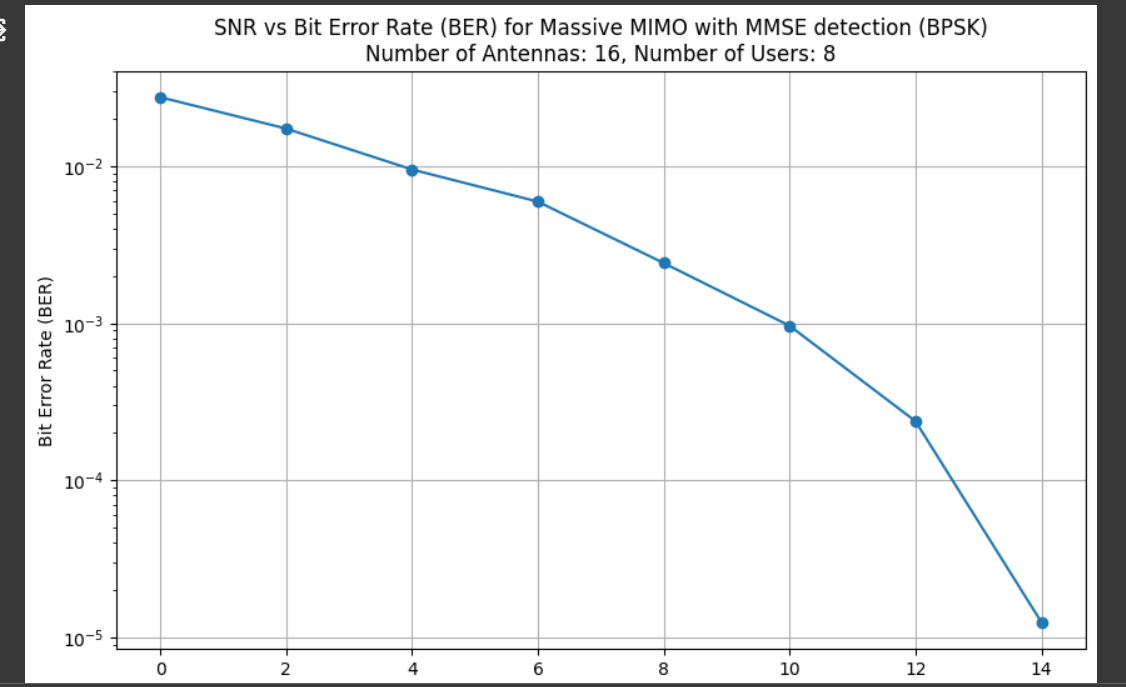
plt.ylabel('Bit Error Rate (BER)')

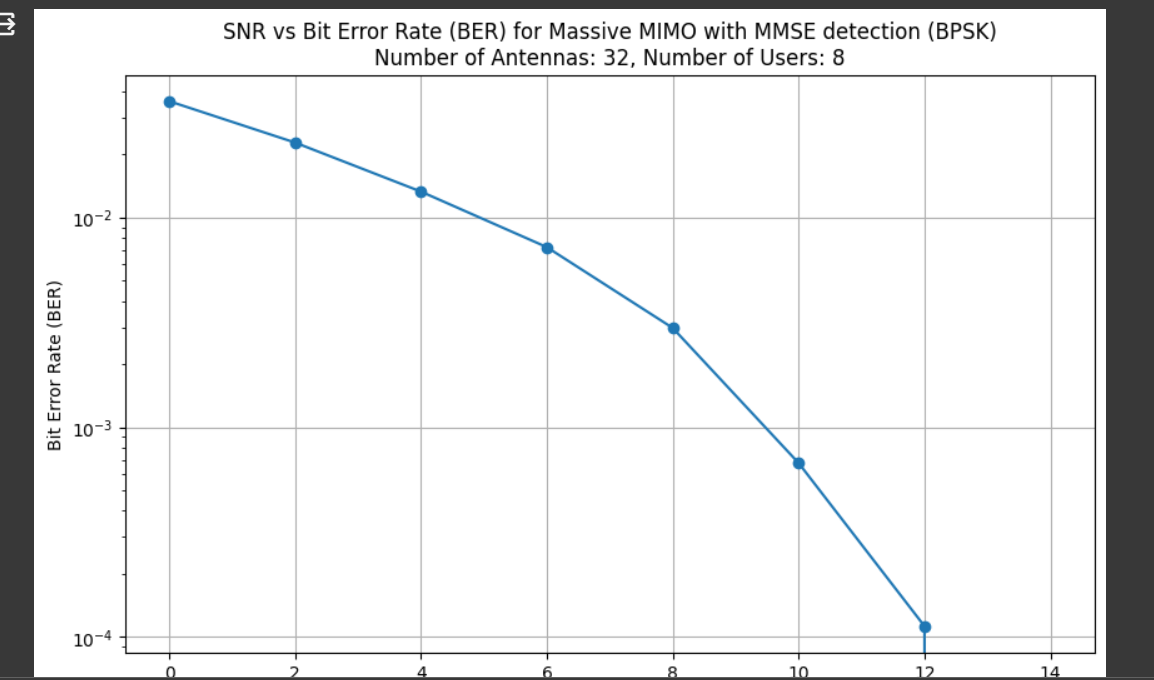
plt.grid(True)

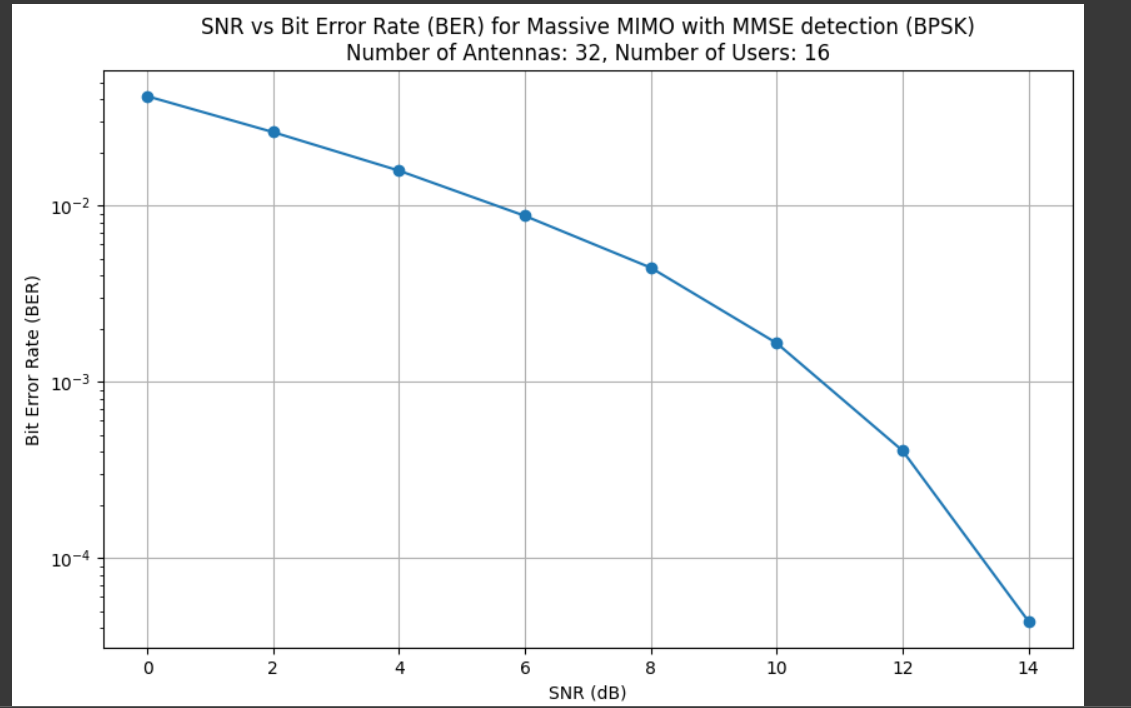
plt.show()

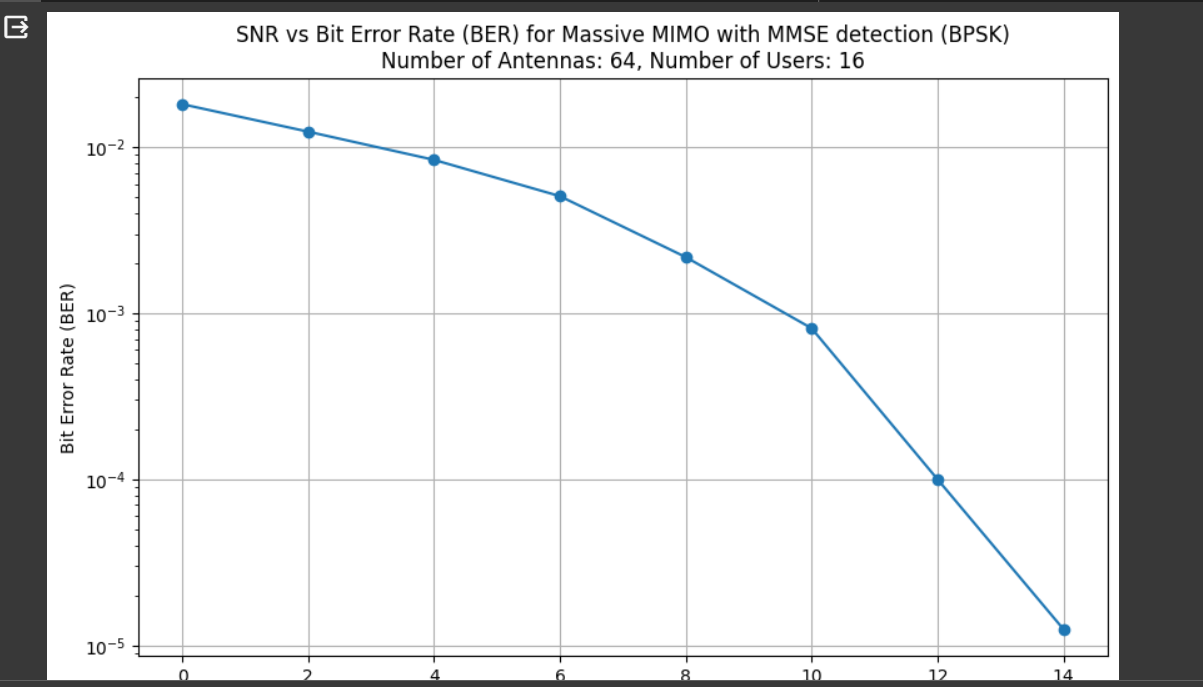
2X4











Detnet

