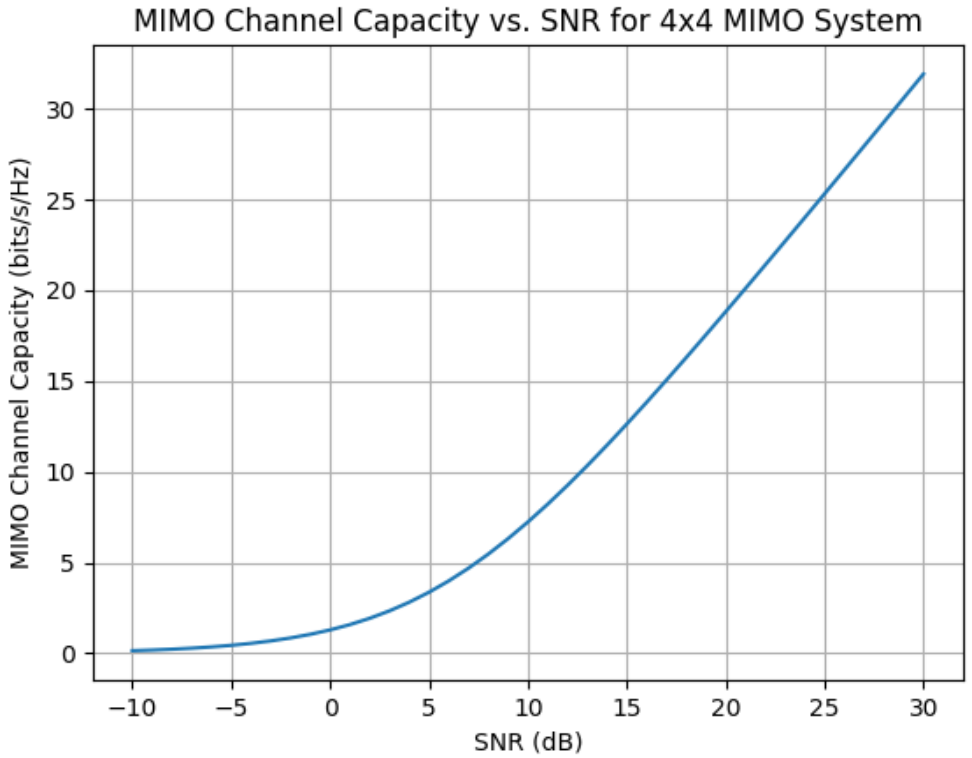
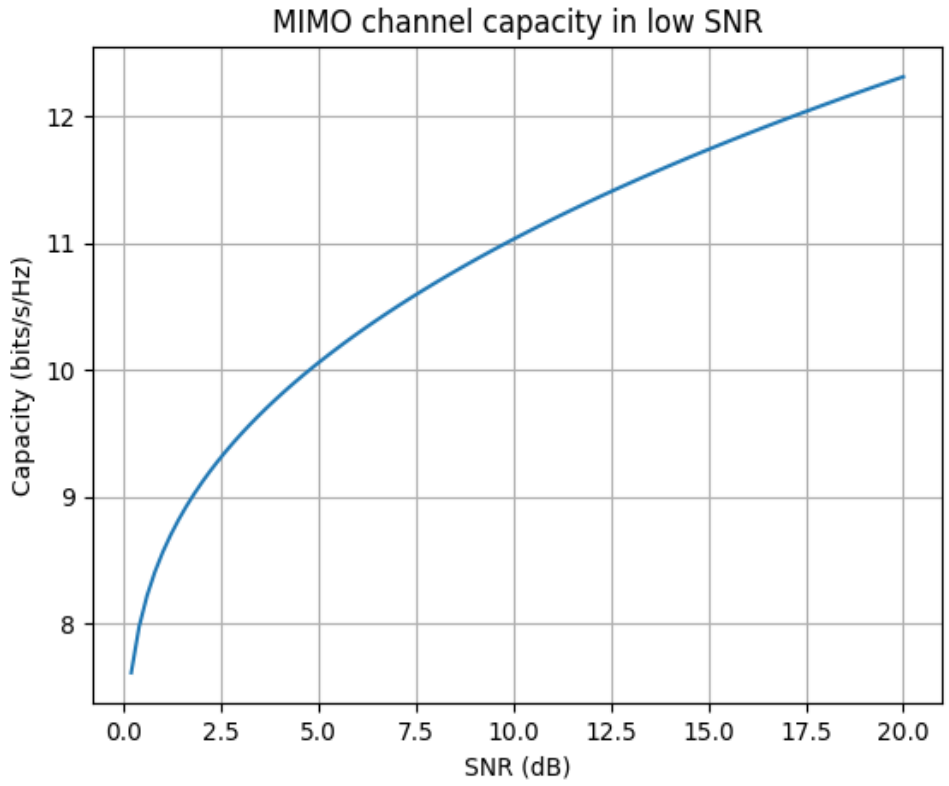
```python
import numpy as np
import matplotlib.pyplot as plt

NT = 4  # number of transmit antennas
NR = 4  # number of receive antennas
q = 1  # fading parameter
m = 1  # shape parameter of the Hoyt distribution

# Define the SNR range in dB
snr_dB = np.arange(-10, 31)

# Convert SNR to linear scale
snr = 10**(snr_dB/10)

# Calculate the MIMO channel capacity in the low SNR regime
C = NT * np.log2(1 + snr/(NT*q/m))

# Plot the MIMO channel capacity versus SNR
plt.plot(snr_dB, C)
plt.xlabel('SNR (dB)')
plt.ylabel('MIMO Channel Capacity (bits/s/Hz)')
plt.title(f'MIMO Channel Capacity vs. SNR for {NT}x{NR} MIMO System')
plt.grid(True)
plt.show()
```

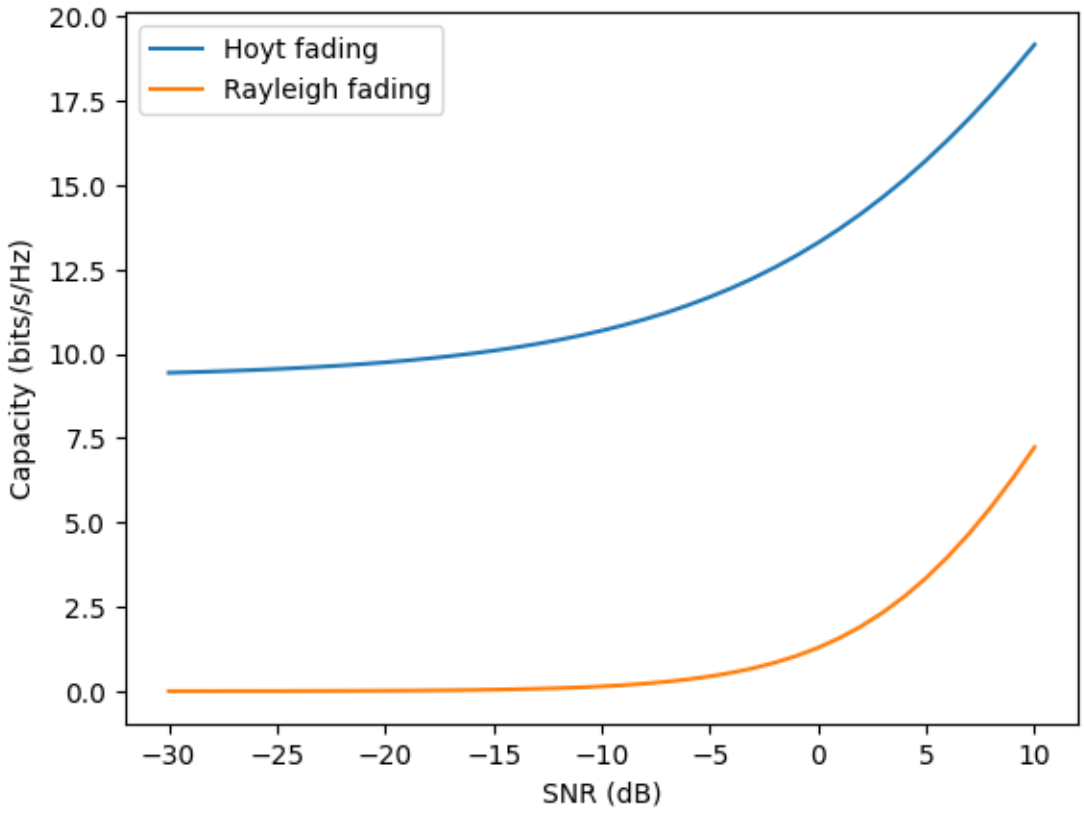


```python
import numpy as np
import matplotlib.pyplot as plt

# Set the parameters
NT = 4 # number of transmit antennas
NR = 2 # number of receive antennas
q = 1 # shape parameter of the Hoyt fading distribution
m = 1 # scale parameter of the Hoyt fading distribution
sigma2 = 1 # noise variance
SNR = np.linspace(0, 20, 100) # range of SNR values to plot

# Compute the MIMO channel capacity in low SNR
lambda_max = np.sqrt(NR/(2*q)) * (1 + np.sqrt((2*q/m) * NR/SNR))**2
C_lsnr = NT * (np.log2(1 + SNR/NR * lambda_max) - np.log2(np.e) * (2*q*(NT
-NR)/(NR*lambda_max)))

# Plot the results
plt.plot(SNR, C_lsnr)
plt.xlabel('SNR (dB)')
plt.ylabel('Capacity (bits/s/Hz)')
plt.title('MIMO channel capacity in low SNR')
plt.grid()
plt.show()
```



**VS PLOTS:**

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

# Define parameters
NT = 4  # number of transmit antennas
NR = 4  # number of receive antennas
q = 0.5  # shape parameter of Hoyt fading
m = 1  # scale parameter of Hoyt fading
sigma2 = 1  # noise variance

# Generate SNR values
SNR_dB = np.linspace(-30, 10, 41)
SNR = 10**(SNR_dB/10)

# Calculate channel capacities
C_lsnr = np.zeros_like(SNR)
C_rayleigh = np.zeros_like(SNR)

for i in range(len(SNR)):
    # Hoyt fading
    lambda_max = np.sqrt(NR/(2*q)) * (1 + np.sqrt((2*q/m) * NR/SNR[i]))**2
    lambda_hoyt = (NR/(2*q)) * (1 + np.sqrt((2*q/m) * NR/SNR[i]))**2
    C_lsnr[i] = NT * (np.log2(1 + SNR[i]/NR * lambda_hoyt) - np.log2(np.e)
 * (2*q*(NT-NR)/(NR*lambda_max)))

    # Rayleigh fading
    C_rayleigh[i] = NR * np.log2(1 + SNR[i]/NR)

# Plot results
plt.figure()
plt.plot(SNR_dB, C_lsnr, label='Hoyt fading')
plt.plot(SNR_dB, C_rayleigh, label='Rayleigh fading')
plt.xlabel('SNR (dB)')
plt.ylabel('Capacity (bits/s/Hz)')
plt.legend()
plt.show()
```




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

# MIMO system parameters
Nt = 4  # number of transmit antennas
Nr = 2  # number of receive antennas

# Simulation parameters
SNR_dB = np.linspace(-20, 20, 41)  # SNR range in dB
SNR = 10 ** (SNR_dB / 10)  # SNR range in linear scale
P = 1  # transmit power
sigma2 = 1  # noise power
q = 1  # shape factor of Hoyt fading
m = 1  # Nakagami-m fading parameter

# Channel capacity calculation
capacity_hoyt = Nt * np.log2(1 + SNR / Nr * (1 + q) / (2 * q))  # Hoyt fad
ing
capacity_nakagami = Nt * np.log2(1 + SNR / Nr * m / (2 * (m + 1)))  # Naka
gami-m fading

# Plotting the results
plt.plot(SNR_dB, capacity_hoyt, label='Hoyt fading')
plt.plot(SNR_dB, capacity_nakagami, label='Nakagami-m fading')
plt.xlabel('SNR (dB)')
```

```
plt.ylabel('Channel capacity (bits/s/Hz)')
plt.title('MIMO channel capacity in low SNR')
plt.legend()
plt.show()
```