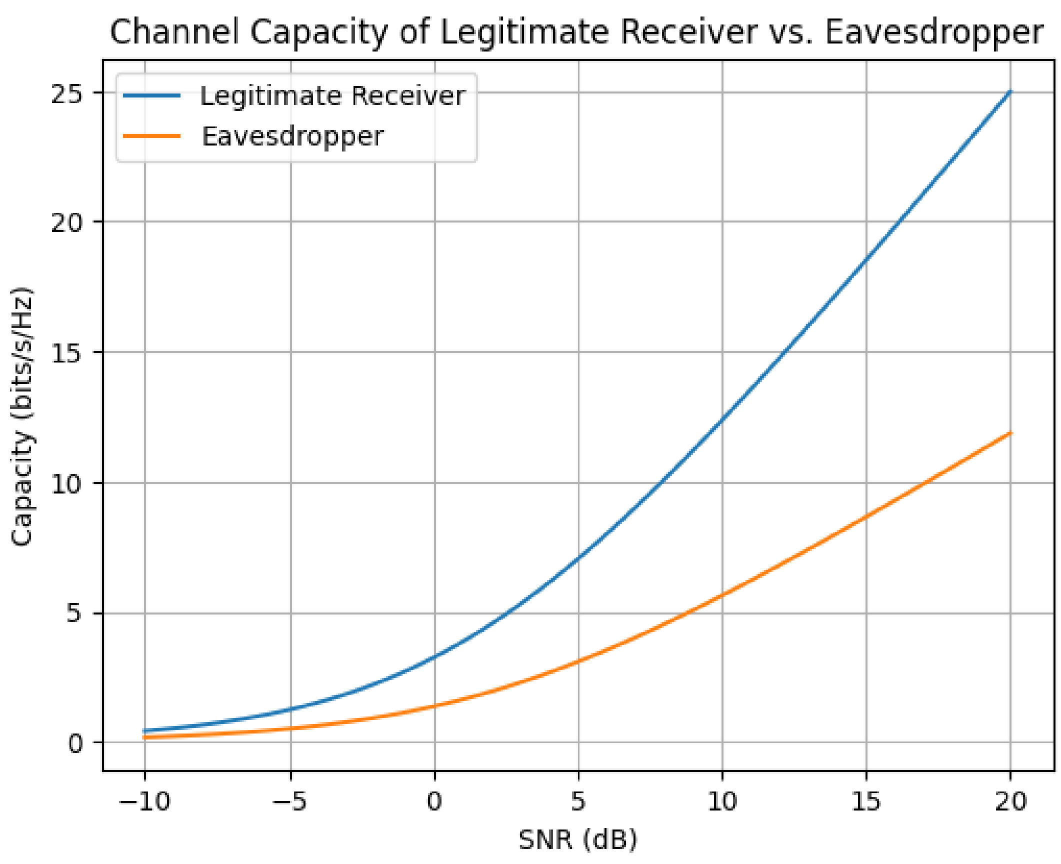
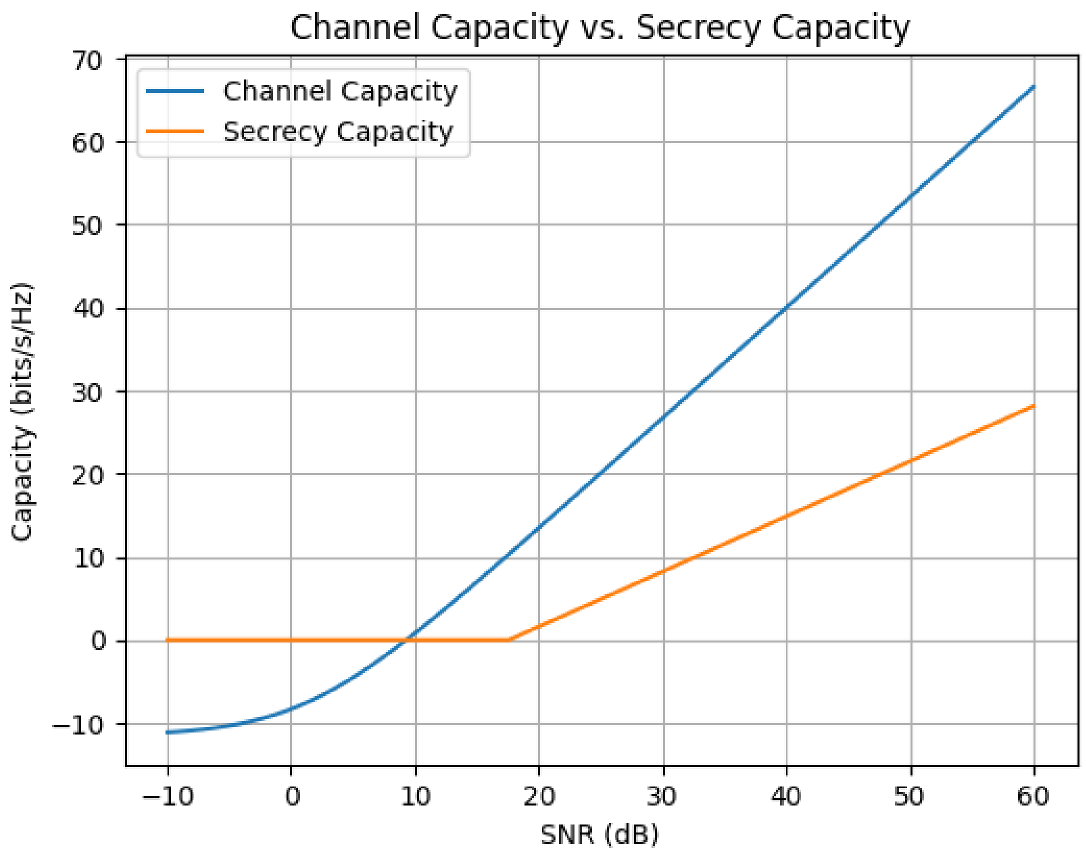
```python
import numpy as np
import matplotlib.pyplot as plt

# Parameters
NT = 4  # Number of transmit antennas
NR = 2  # Number of receive antennas
q = 1.5  # Shape parameter of Hoyt fading
SNR_dB = np.linspace(-10, 20, 100)  # SNR range in dB

# Conversion from dB to linear scale
SNR = 10**(SNR_dB / 10)

# Average channel gains
lambda_B = 1.5
lambda_E = 1.2

# Calculate channel capacity and eavesdropper capacity
C_B = NT * np.log2(1 + SNR/NR * lambda_B)
C_E = NR * np.log2(1 + SNR/NR * lambda_E)

# Plot channel capacity of the legitimate receiver and eavesdropper
plt.plot(SNR_dB, C_B, label='Legitimate Receiver')
plt.plot(SNR_dB, C_E, label='Eavesdropper')
plt.xlabel('SNR (dB)')
plt.ylabel('Capacity (bits/s/Hz)')
plt.title('Channel Capacity of Legitimate Receiver vs. Eavesdropper')
plt.legend()
plt.grid(True)
plt.show()
```



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

# Parameters
NT = 4  # Number of transmit antennas
NR = 2  # Number of receive antennas
q = 1.5  # Shape parameter of Hoyt fading
SNR_dB = np.linspace(-10, 60, 100)  # SNR range in dB

# Conversion from dB to linear scale
SNR = 10**(SNR_dB / 10)

# Average channel gains
lambda_B = 1.5
lambda_E = 1.2

# Calculate channel capacity and secrecy capacity
C_lsnr = NT * (np.log2(1 + SNR/NR * lambda_B) - np.log2(np.e) * (2*q*(NT-
NR)/(NR*lambda_B)))
C_eve = NR * np.log2(1 + SNR/NR * lambda_E)
C_sec = np.maximum(C_lsnr - C_eve, 0)

# Plot channel capacity and secrecy capacity
plt.plot(SNR_dB, C_lsnr, label='Channel Capacity')
plt.plot(SNR_dB, C_sec, label='Secrecy Capacity')
plt.xlabel('SNR (dB)')
plt.ylabel('Capacity (bits/s/Hz)')
plt.title('Channel Capacity vs. Secrecy Capacity')
plt.legend()
plt.grid(True)
plt.show()
```


Channel Capacity vs. Secrecy Capacity
Channel Capacity
Secrecy Capacity
Capacity (bits/s/Hz)
SNR (dB)