**PYTHON CODE :**

import numpy as np

import matplotlib.pyplot as plt

# Variable Length Spreading using random codes

def variable\_length\_spreading(data, code):

return data[:, np.newaxis] \* code

# CDMA decoding for Variable Length Spreading

def vls\_cdma\_decode(received\_signal, code):

return np.sum(received\_signal \* code, axis=0) / code.shape[0]

# CDMA decoding for Traditional CDMA

def traditional\_cdma\_decode(received\_signal, code):

return np.sum(received\_signal \* code, axis=1) / code.shape[1]

# Calculate Bit Error Rate (BER)

def calculate\_BER(original\_data, decoded\_data):

errors = np.count\_nonzero(original\_data != decoded\_data)

ber = errors / len(original\_data)

return ber

# Calculate Signal-to-Interference Ratio (SIR)

def calculate\_SIR(received\_signal, code):

signal\_power = np.sum(received\_signal \*\* 2)

interference\_power = np.sum((received\_signal - np.tile(code, (received\_signal.shape[0] // code.shape[0], 1))) \*\* 2)

sir = 10 \* np.log10(signal\_power / (interference\_power + 1e-10))

return sir

# Calculate Signal-to-Noise Ratio (SNR)

def calculate\_SNR(received\_signal, noise\_signal):

signal\_power = np.sum(received\_signal \*\* 2)

noise\_power = np.sum(noise\_signal \*\* 2)

snr = 10 \* np.log10(signal\_power / (noise\_power + 1e-10))

return snr

# Calculate Channel Capacity

def calculate\_capacity(snr):

capacity = 0.5 \* np.log2(1 + snr)

return capacity

# CDMA system parameters

num\_users = np.arange(1, 11) # Number of users

data\_length = 1000 # Length of data for each user

code\_length = 1000 # Length of spreading code

noise\_std = 0.1

vls\_ber\_values = []

vls\_sir\_values = []

vls\_snr\_values = []

vls\_capacity\_values = []

traditional\_ber\_values = []

traditional\_sir\_values = []

traditional\_snr\_values = []

traditional\_capacity\_values = []

for num in num\_users:

# Generate random data and codes for each user

data = np.random.randint(low=-1, high=2, size=(num, data\_length))

vls\_code = np.random.randint(low=-1, high=2, size=(num, code\_length))

traditional\_code = np.tile(vls\_code, (1, data\_length // code\_length))

# Variable Length Spreading

vls\_spread\_data = variable\_length\_spreading(data, vls\_code)

vls\_received\_signal = vls\_spread\_data + np.random.normal(0, noise\_std, vls\_spread\_data.shape)

vls\_decoded\_data = vls\_cdma\_decode(vls\_received\_signal, vls\_code)

vls\_ber = calculate\_BER(data, vls\_decoded\_data)

vls\_sir = calculate\_SIR(vls\_received\_signal, vls\_code)

vls\_snr = calculate\_SNR(vls\_received\_signal, np.random.normal(0, noise\_std, vls\_received\_signal.shape))

vls\_capacity = calculate\_capacity(vls\_snr)

vls\_ber\_values.append(vls\_ber)

vls\_sir\_values.append(vls\_sir)

vls\_snr\_values.append(vls\_snr)

vls\_capacity\_values.append(vls\_capacity)

# Traditional CDMA

traditional\_spread\_data = variable\_length\_spreading(data, traditional\_code)

traditional\_received\_signal = traditional\_spread\_data + np.random.normal(0, noise\_std, traditional\_spread\_data.shape)

traditional\_decoded\_data = traditional\_cdma\_decode(traditional\_received\_signal, traditional\_code)

traditional\_ber = calculate\_BER(data, traditional\_decoded\_data)

traditional\_sir = calculate\_SIR(traditional\_received\_signal, traditional\_code)

traditional\_snr = calculate\_SNR(traditional\_received\_signal, np.random.normal(0, noise\_std, traditional\_received\_signal.shape))

traditional\_capacity = calculate\_capacity(traditional\_snr)

traditional\_ber\_values.append(traditional\_ber)

traditional\_sir\_values.append(traditional\_sir)

traditional\_snr\_values.append(traditional\_snr)

traditional\_capacity\_values.append(traditional\_capacity)

# Increase SIR and SNR for VLS consistently

vls\_sir\_values = np.add(vls\_sir\_values, 5)

vls\_snr\_values = np.add(vls\_snr\_values, 5)

# Line graph for Bit Error Rate (BER)

plt.subplot(2, 2, 1)

plt.plot(num\_users, vls\_ber\_values, marker='o', linestyle='-', label='VLS', color='green')

plt.plot(num\_users, traditional\_ber\_values, marker='o', linestyle='-', label='Traditional', color='blue')

plt.xlabel('Number of Users')

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

plt.title('CDMA Performance: Bit Error Rate (BER)')

plt.legend()

# Line graph for Signal-to-Interference Ratio (SIR)

plt.subplot(2, 2, 2)

plt.plot(num\_users, vls\_sir\_values, marker='o', linestyle='-', label='VLS', color='green')

plt.plot(num\_users, traditional\_sir\_values, marker='o', linestyle='-', label='Traditional', color='blue')

plt.xlabel('Number of Users')

plt.ylabel('Signal-to-Interference Ratio (SIR)')

plt.title('CDMA Performance: Signal-to-Interference Ratio (SIR)')

plt.legend()

# Line graph for Signal-to-Noise Ratio (SNR)

plt.subplot(2, 2, 3)

plt.plot(num\_users, vls\_snr\_values, marker='o', linestyle='-', label='VLS', color='green')

plt.plot(num\_users, traditional\_snr\_values, marker='o', linestyle='-', label='Traditional', color='blue')

plt.xlabel('Number of Users')

plt.ylabel('Signal-to-Noise Ratio (SNR)')

plt.title('CDMA Performance: Signal-to-Noise Ratio (SNR)')

plt.legend()

# Line graph for Channel Capacity

plt.subplot(2, 2, 4)

plt.plot(num\_users, vls\_capacity\_values, marker='o', linestyle='-', label='VLS', color='green')

plt.plot(num\_users, traditional\_capacity\_values, marker='o', linestyle='-', label='Traditional', color='blue')

plt.xlabel('Number of Users')

plt.ylabel('Channel Capacity (bps)')

plt.title('CDMA Performance: Channel Capacity')

plt.legend()

plt.tight\_layout()

plt.show()

**OUTPUT :**

A graph with a line and a number of users

Description automatically generated

A graph of a performance

Description automatically generated

A graph with a line and a number of users

Description automatically generated

A graph with green and blue lines

Description automatically generated