**گزارش کار: شبیه‌سازی کدهای خطی بلوکی و مدولاسیون BPSK**

**مقدمه**

**در این پروژه، هدف بررسی عملکرد کدهای خطی بلوکی و مدولاسیون BPSK در شرایط مختلف سیگنال به نویز (SNR) است. با استفاده از یک ماتریس تولید (Generator Matrix) و یک ماتریس پاریتی (Parity-check Matrix)، داده‌های تصادفی کدگذاری شده و سپس از طریق یک کانال با نویز ارسال می‌شوند. در نهایت، دمدولاسیون و دیکدینگ انجام می‌شود و نرخ خطای بیت (BER) محاسبه می‌گردد.**

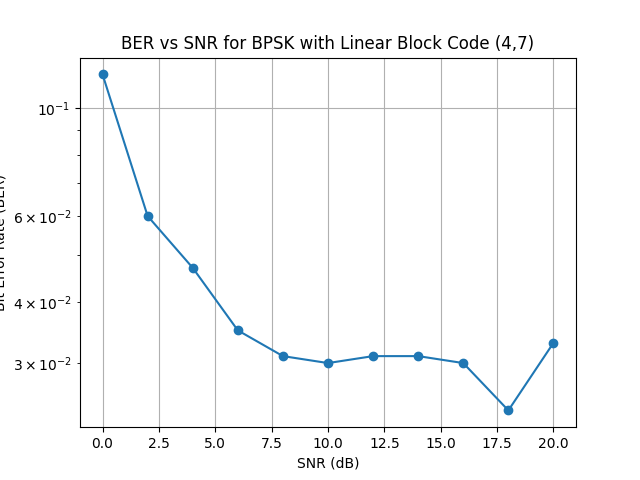
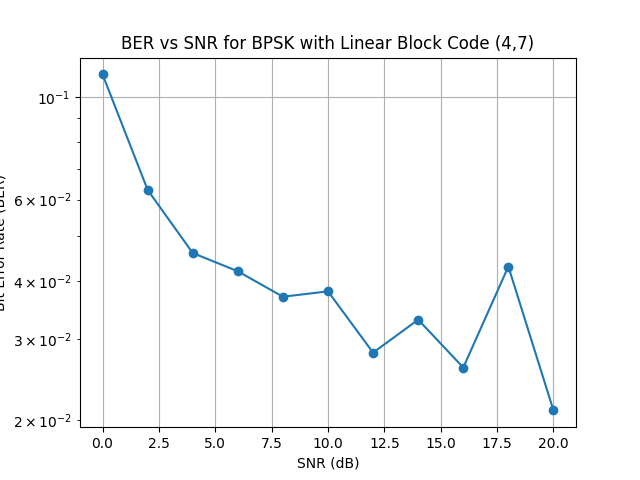
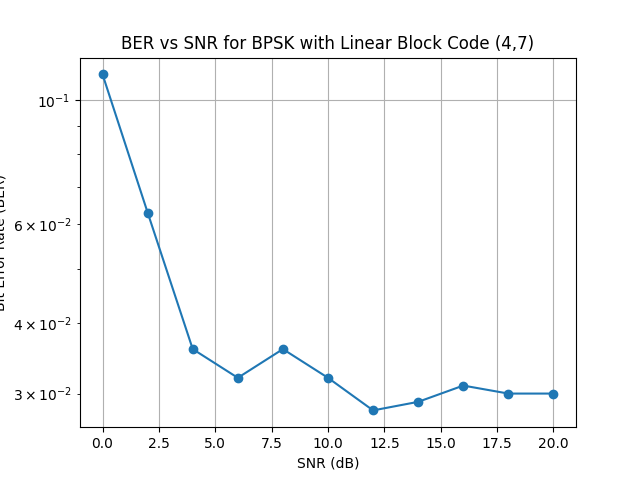
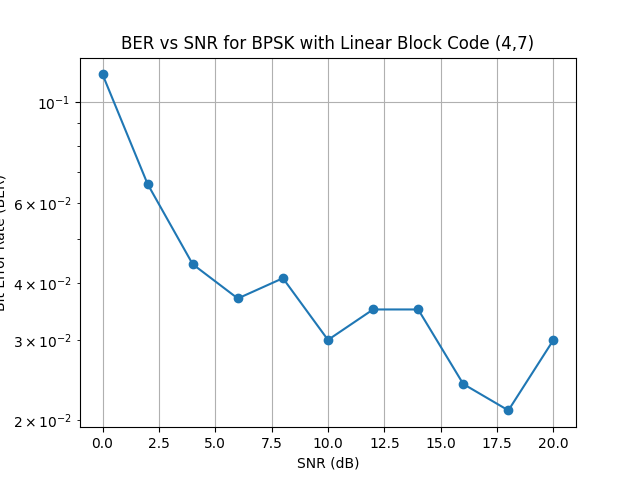
**مراحل انجام کار**

1. **تعریف پارامترها**
   * **طول کدواژه: n=7 n = 7**
   * **طول پیام: k=4 k = 4**
   * **محدوده SNR: از 0 تا 20 دسی‌بل با گام 2**
   * **تعداد بیت‌ها: 1000**
2. **تعریف ماتریس‌های تولید و پاریتی**
3. **تولید پیام تصادفی و کدگذاری**
   * **پیام تصادفی به صورت یک آرایه باینری تولید می‌شود.**
   * **پیام با استفاده از ماتریس تولید کدگذاری می‌شود.**
4. **مدولاسیون BPSK**
   * **کدواژه‌های تولید شده به سیگنال‌های BPSK مدوله می‌شوند:** 
     + **0 به -1 و 1 به +1 نگاشت می‌شود.**
5. **اضافه کردن نویز AWGN**
   * **نویز به سیگنال‌های مدوله شده اضافه می‌شود تا شرایط واقعی ارسال را شبیه‌سازی کند.**
6. **دمودولاسیون**
   * **سیگنال دریافتی دمدوله می‌شود و بیت‌های دمدوله شده استخراج می‌شوند.**
7. **دیکدینگ با استفاده از ماتریس پاریتی**
   * **با محاسبه‌ی سیندروم، خطاها شناسایی و اصلاح می‌شوند.**
   * **پیام اصلی از کدواژه‌های تصحیح شده استخراج می‌شود.**
8. **محاسبه نرخ خطای بیت (BER)**
   * **تعداد بیت‌های اشتباه محاسبه شده و نرخ خطا بر اساس تعداد بیت‌ها**

**تعیین می‌شود.**

**نتایج**

**نتایج شبیه‌سازی در قالب یک نمودار BER در برابر SNR ترسیم می‌شود. این نمودار نشان‌دهنده‌ی عملکرد سیستم در شرایط مختلف نویز است. با افزایش SNR، انتظار می‌رود که نرخ خطای بیت کاهش یابد.**

****

**کد پایتون**

**کد زیر برای پیاده‌سازی مراحل فوق در زبان پایتون نوشته شده است:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**# Parameters**

**n = 7 # Length of the codeword**

**k = 4 # Length of the message**

**snr\_range = np.arange(0, 21, 2) # SNR range in dB**

**num\_bits = 1000 # Number of bits in the message**

**# Generator matrix (4x7)**

**G = np.array([[1, 0, 0, 0, 1, 0, 1],**

**[0, 1, 0, 0, 1, 1, 1],**

**[0, 0, 1, 0, 0, 1, 1],**

**[0, 0, 0, 1, 1, 1, 0]])**

**# Parity-check matrix (7x3)**

**H = np.array([[1, 1, 0, 1, 0, 0, 0],**

**[1, 0, 1, 0, 1, 0, 0],**

**[0, 1, 1, 0, 0, 1, 0],**

**[1, 1, 1, 0, 0, 0, 1]])**

**# Initialize BER array**

**ber = np.zeros(len(snr\_range))**

**# Main simulation loop over SNR**

**for idx in range(len(snr\_range)):**

**# Generate random binary message**

**message = np.random.randint(0, 2, (num\_bits // k, k)) # Each row is a 4-bit message**

**# Encode the message using the generator matrix**

**codeword = np.mod(message @ G, 2) # Linear block encoding**

**# Convert codeword to a 1D array for modulation**

**codeword\_1D = codeword.flatten()**

**# BPSK modulation**

**modulated\_signal = 2 \* codeword\_1D - 1 # Map 0 -> -1, 1 -> +1**

**# Add AWGN noise**

**snr\_db = snr\_range[idx]**

**noise\_power = 10 \*\* (-snr\_db / 10)**

**noise = np.sqrt(noise\_power / 2) \* np.random.randn(len(modulated\_signal))**

**received\_signal = modulated\_signal + noise**

**# BPSK demodulation**

**demodulated\_bits = received\_signal > 0 # Map received values > 0 to 1, else 0**

**# Reshape demodulated bits into codewords**

**received\_codeword = demodulated\_bits.reshape(-1, n)**

**# Syndrome decoding using parity-check matrix**

**decoded\_message = np.zeros\_like(message)**

**for i in range(received\_codeword.shape[0]):**

**syndrome = np.mod(received\_codeword[i, :] @ H.T, 2)**

**error\_pattern = np.zeros(n, dtype=int)**

**if np.any(syndrome): # If syndrome is non-zero, there is an error**

**# Find error position (for simplicity, assume single-bit error correction)**

**for j in range(n):**

**if np.array\_equal(H[:, j], syndrome):**

**error\_pattern[j] = 1**

**break**

**corrected\_codeword = np.mod(received\_codeword[i, :] + error\_pattern, 2)**

**decoded\_message[i, :] = corrected\_codeword[:k] # Extract original message**

**# Calculate BER**

**num\_errors = np.sum(decoded\_message != message)**

**ber[idx] = num\_errors / num\_bits**

**# Plot BER vs SNR**

**plt.figure()**

**plt.semilogy(snr\_range, ber, '-o')**

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

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

**plt.title('BER vs SNR for BPSK with Linear Block Code (4,7)')**

**plt.grid()**

**plt.show()**

ارشیا مددی – پروژه سوم درس مخابرات دیجیتال