Homework 5

06 November 2023 20:50

5.6

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

def convert_symbol_to_input(c):
    return np.array([1 - 2 * i for i in c])

def bawgn(c, sigma2):
    x = convert_symbol_to_input(c)
    return np.random.normal(x, np.sqrt(sigma2))

def eb_snr_db_to_noise_var(snr, rate):
    return 1 / (2 * (10 ** (snr / 10)) * rate)

n_mcmc = int(1e5)
eb_snr_db_list = [i for i in range(1, 13)]
max_iter = 5
H = np.array([
    [1, 1, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 1, 1, 1, 1],
    [1, 0, 0, 1, 0, 1, 0, 1],
    [0, 1, 1, 0, 1, 0, 1, 0]
])
m, n = H.shape
c = [0] * n
rate = 1 - (np.linalg.matrix_rank(H) / n)
M = np.array([[r for r in range(m) if H[r, c] == 1] for c in range(n)])
N = np.array([[c for c in range(n) if H[r, c] == 1] for r in range(m)])
Q = np.zeros_like(N)
R = np.zeros_like(M)
wer_list, ber_list = [], []
for snr in eb_snr_db_list:
    sigma2 = eb_snr_db_to_noise_var(snr, rate)
    wer = ber = 0
    for _ in range(n_mcmc):
        y = bawgn(c, sigma2)
        L = (2 / sigma2) * y
        Q = np.array([[L[j] if j in N[i] else None for j in range(n)] for i in range(m)])
        n_iter = 0
        while n_iter < max_iter:
            R = np.array([
                [2 * np.arctanh(np.prod([np.tanh(Q[j, k] / 2)
                                         for k in range(n) if k != i and Q[j, k] != None]))
                 if j in M[i] else 0 for j in range(m)] for i in range(n)
            ])
            Q = np.array([
                    [L[j] + sum([R[j, k] for k in range(m) if k != i])
                     if j in N[i] else None for j in range(n)] for i in range(m)
                ])
            t = [L[i] + sum(R[i]) for i in range(n)]
            c_hat = np.array([0 if t[i] >= 0 else 1 for i in range(n)])
            if np.array_equal((c_hat.dot(H.T) % 2), np.zeros(m)):
                break
            n_iter += 1
        if not np.array_equal(c, c_hat):
            wer += 1
            ber += sum([1 for i, j in zip(c, c_hat) if i != j])
    wer_list.append(wer / n_mcmc)
    ber_list.append(ber / (n * n_mcmc))
    print(f"SNR/Sigma2: {snr}/{sigma2:.04} - WER: {wer / n_mcmc:0.4} - BER: {ber / (n *
n_mcmc):0.4}")
print(wer_list)
print(ber_list)
plt.plot(eb_snr_db_list, wer_list, color='blue')
plt.plot(eb_snr_db_list, ber_list, color='red')
plt.xlabel("Eb / N0 SNR dB")
plt.legend(["WER", "BER"])
plt.yscale('log')
plt.savefig('5.6.jpg')
```