Homework 4

27 October 2023 15:37

4.1

a) $y_1 = [0, 0, 1, x_1, x_2, x_3]$

$$Hy_1 = \begin{bmatrix} x_1 \\ x_2 + 1 \\ x_2 + x_3 \\ x_1 + x_3 + 1 \end{bmatrix} = 0 \Leftrightarrow \begin{cases} x_1 = 0 \\ x_2 = 1 \\ x_3 = 1 \end{cases} . \quad \hat{c}_1 = [0, 0, 1, 0, 1, 1]$$

b) $y_2 = [x_1, x_2, x_3, 1, 0, 1]$

$$Hy_2 = \begin{bmatrix} x_1 + x_2 + 1 \\ x_2 + x_3 \\ x_1 + 1 \\ x_3 + 1 + 1 \end{bmatrix} = 0 \Leftrightarrow \begin{cases} x_1 = 1 \\ x_2 = 0 \\ x_3 = 0 \end{cases} . \quad \hat{c}_2 = [1, 0, 0, 1, 0, 1]$$

c) $y_3 = [x_1, x_2, x_3, 0, 1, x_4]$

$$Hy_3 = \begin{bmatrix} x_1 + x_2 \\ x_2 + x_3 + 1 \\ x_1 + x_4 + 1 \\ x_3 + x_4 \end{bmatrix} = 0 \Leftrightarrow \begin{cases} x_2 = x_1 \\ x_3 = 1 - x_1 \\ x_4 = 1 - x_1 \end{cases} . \quad \text{Failure to decode}$$

4.2

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

def bsc(u, p):
    return [i if np.random.choice([0, 1], p=[p, 1 - p]) == 1 else (1 - i)
            for i in u]
H = np.array([
    [1, 0, 1, 1, 1, 0, 0],
    [1, 1, 0, 1, 0, 1, 0],
    [1, 1, 1, 0, 0, 0, 1]
])
G = np.array([
    [1, 0, 0, 0, 1, 1, 1],
    [0, 1, 0, 0, 0, 1, 1],
    [0, 0, 1, 0, 1, 0, 1],
    [0, 0, 0, 1, 1, 1, 0]
])
syndrome_tables = {
    (0, 0, 0): np.array([0, 0, 0, 0, 0, 0, 0]),
    (0, 0, 1): np.array([0, 0, 0, 0, 0, 0, 1]),
    (0, 1, 0): np.array([0, 0, 0, 0, 0, 1, 0]),
    (0, 1, 1): np.array([0, 1, 0, 0, 0, 0, 0]),
    (1, 0, 0): np.array([0, 0, 0, 0, 1, 0, 0]),
    (1, 0, 1): np.array([0, 0, 1, 0, 0, 0, 0]),
    (1, 1, 0): np.array([0, 0, 0, 1, 0, 0, 0]),
    (1, 1, 1): np.array([1, 0, 0, 0, 0, 0, 0])
}
N = int(1e4)
p_list = [0.1, 0.025, 0.0025]
for p in p_list:
    wer = 0
    for _ in range(N):
        u = np.random.choice([0, 1], size=4)
        c = u.dot(G) % 2
        y = bsc(c, p)
        s = tuple((H.dot(y) % 2).tolist())
        e = syndrome_tables[s]
        c_hat = (y + e) % 2
        if not np.array_equal(c, c_hat):
            wer += 1
    print(f"P: {p} - WER: {wer / N}")
```

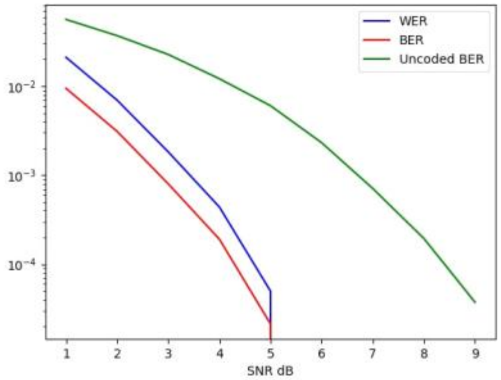
```
P: 0.1 - WER: 0.1513
P: 0.025 - WER: 0.0114
P: 0.0025 - WER: 0.0002
```

4.3

```python
import itertools
```

4.3

```
import itertools
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(i, np.sqrt(sigma2)) for i in x]

def snr_db_to_noise_var(snr):
    return (1 / (10 ** (snr / 10))) / 2

G = np.array([
    [1, 0, 0, 0, 1, 1, 1],
    [0, 1, 0, 0, 0, 1, 1],
    [0, 0, 1, 0, 1, 0, 1],
    [0, 0, 0, 1, 1, 1, 0]
])
U = np.array(list(itertools.product([0, 1], repeat=4)))
C = [
    np.sum(np.array(u)[:, None] * G, axis=0) % 2 for u in U
]
N = int(1e5)
snr_db_list = [i for i in range(1, 10)]
wer_list, ber_list, uncoded_ber_list = [], [], []
for snr in snr_db_list:
    sigma2 = snr_db_to_noise_var(snr)
    wer = ber = uncoded_ber = 0
    for _ in range(N):
        u = np.random.choice([0, 1], size=4)
        c = u.dot(G) % 2
        y = bawgn(c, sigma2)
        diffs = [
            np.sum(np.power(convert_symbol_to_input(i) - y, 2),
axis=-1) for i in C
        ]
        c_hat = C[np.argsort(diffs)[0]]
        if not np.array_equal(c, c_hat):
            wer += 1
            ber += sum([1 for i, j in zip(c, c_hat) if i != j])

        y = bawgn(u, sigma2)
        diffs = [
            np.sum(np.power(convert_symbol_to_input(i) - y, 2),
axis=-1) for i in U
        ]
        u_hat = U[np.argsort(diffs)[0]]
        if not np.array_equal(u, u_hat):
            uncoded_ber += sum([1 for i, j in zip(u, u_hat) if i !=
j])
    wer_list.append(wer / N)
    ber_list.append(ber / (7 * N))
    uncoded_ber_list.append(uncoded_ber / (4 * N))
    print(f"SNR/Sigma2: {snr}/{sigma2:.04} - WER: {wer / N:0.4} - "
          f"BER: {ber / (7 * N):0.4} - Uncoded BER: {uncoded_ber /
(4 * N):0.4}")
print(wer_list)
print(ber_list)
plt.plot(snr_db_list, wer_list, color='blue')
plt.plot(snr_db_list, ber_list, color='red')
plt.plot(snr_db_list, uncoded_ber_list, color='green')
plt.xlabel("SNR dB")
plt.legend(["WER", "BER", "Uncoded BER"])
plt.yscale('log')
plt.savefig('4.3.jpg')
```



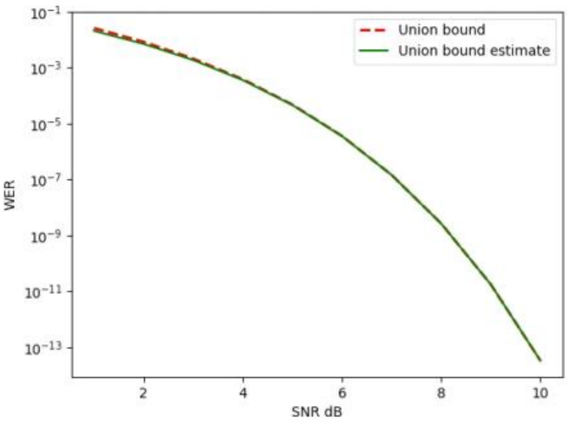
4.4

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

def snr_db_to_noise_var(snr):
    return (1 / (10 ** (snr / 10))) / 2

G = np.array([
    [1, 0, 0, 0, 1, 1, 1],
    [0, 1, 0, 0, 0, 1, 1],
    [0, 0, 1, 0, 1, 0, 1],
    [0, 0, 0, 1, 1, 1, 0]
])
U = np.array(list(itertools.product([0, 1], repeat=4)))
C = [
    np.sum(np.array(u)[:, None] * G, axis=0) % 2 for u in U
]
weight_dict = {
```

```
SNR/Sigma2: 1/0.3972 - Union bound: 0.02625 -
Union bound estimate: 0.02096
SNR/Sigma2: 2/0.3155 - Union bound: 0.00845 -
Union bound estimate: 0.007155
SNR/Sigma2: 3/0.2506 - Union bound: 0.002117 -
Union bound estimate: 0.001891
SNR/Sigma2: 4/0.1991 - Union bound: 0.0003881 -
Union bound estimate: 0.0003623
SNR/Sigma2: 5/0.1581 - Union bound: 4.811e-05 -
Union bound estimate: 4.639e-05
SNR/Sigma2: 6/0.1256 - Union bound: 3.635e-06 -
Union bound estimate: 3.577e-06
SNR/Sigma2: 7/0.09976 - Union bound: 1.466e-07 -
Union bound estimate: 1.458e-07
SNR/Sigma2: 8/0.07924 - Union bound: 2.668e-09 -
Union bound estimate: 2.664e-09
SNR/Sigma2: 9/0.06295 - Union bound: 1.775e-11 -
Union bound estimate: 1.775e-11
```

```
])
U = np.array(list(itertools.product([0, 1], repeat=4)))
C = [
    np.sum(np.array(u)[:, None] * G, axis=0) % 2 for u in U
]
weight_dist = {
    w: len([c for c in C if np.sum(c) == w]) for w in range(1, 8)
}
weight_dist = list(filter(lambda x: x[1] != 0, sorted(weight_dist.items(),
key=lambda x: x[0])))
N = int(1e5)
snr_db_list = [i for i in range(1, 11)]
union_bound_list, union_bound_est_list = [], []
for snr in snr_db_list:
    sigma2 = snr_db_to_noise_var(snr)
    union_bound_list.append(
        0.5 * np.sum([c * scipy.special.erfc(np.sqrt(w / (2 * sigma2)))
                      for w, c in weight_dist])
    )
    union_bound_est_list.append(
        0.5 * weight_dist[0][1] * scipy.special.erfc(np.sqrt(weight_dist[0][0] /
(2 * sigma2))))
    print(f"SNR/Sigma2: {snr}/{sigma2:.04} - Union bound:
{union_bound_list[-1]:0.4} - "
          f"Union bound estimate: {union_bound_est_list[-1]:0.4}")

plt.plot(snr_db_list, union_bound_list, '--', linewidth=2, color='red')
plt.plot(snr_db_list, union_bound_est_list, color='green')
plt.xlabel("SNR dB")
plt.ylabel("WER")
plt.legend(["Union bound", "Union bound estimate"])
plt.yscale('log')
plt.savefig('4.4.jpg')
```

```
SNR/Sigma2: 8/0.07924 - Union bound: 2.668e-09 -
Union bound estimate: 2.664e-09
SNR/Sigma2: 9/0.06295 - Union bound: 1.775e-11 -
Union bound estimate: 1.775e-11
SNR/Sigma2: 10/0.05 - Union bound: 3.32e-14 -
Union bound estimate: 3.32e-14
```


10^{-1}
10^{-3}
10^{-5}
10^{-7}
10^{-9}
10^{-11}
10^{-13}
WER
Union bound
Union bound estimate
SNR dB