Fathan Ananta Nur
236-D8749

44135 Applied Information Theory
Assignment 4 (Revision)

Perform computer simulations of digital communications using (4, 3)-single parity check codes. Compare the probabilities of undetected errors for same p.

1. Memoryless errors (skew Bernoulli map)
   In this experiment, the parameters below will be used:

```
Length (N) = 1000000
Parameter c and t = 0.49999
List of p = {0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.49999}
Initial value = 0.1782612
Initial error value = 0.5673244
```

   The experiment will yield results for both computed and theoretical values of the probability of undetected error, based on the parameters described above and increments in the value of p. The theoretical values will be calculated using the following formula.

$$P_u = 6 \cdot p^2(1-p)^2 + p^4$$

   The experiment was conducted in a Python environment, and the results are displayed in the following table.

Table 1. Probability of Undetected Error on Memoryless Source Bernoulli Map

p	Computed	Theoretical
0.05	0.01357	0.01354
0.1	0.04868	0.04870
0.15	0.09788	0.09804
0.2	0.15513	0.15520
0.25	0.21470	0.21484
0.3	0.27268	0.27270
0.35	0.32623	0.32554
0.4	0.37073	0.37120
0.45	0.40847	0.40854
0.49999	0.43675	0.43749

   Table 1 presents both the computed and theoretical values of the probability of undetected error as a function of incremental values of p. The observed alignment between computed and theoretical values indicates minimal discrepancy. Moreover, an increase in the value of p correlates with a rise in the probability of undetected errors. This trend arises because **higher values of p enhance the likelihood of multiple bit errors occurring simultaneously**. Given that single parity check codes are only capable of detecting an odd number of errors, this increase in simultaneous bit errors, which single parity check codes fail to detect, contributes to a greater overall probability of undetected errors.

2. Markov-type errors (PLM3 map)
In this experiment, the parameters below will be used:

```
Length (N) = 1000000
Parameter c and t = 0.49999
List of p = {0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.49999}
List of another p (p2) = {0.16, 0.34}
Initial value = 0.1782612
Initial error value = 0.5673244
```

The experiment will produce results for both computed and theoretical values of the probability of undetected error, influenced by the specified parameters and incremental adjustments to the value of p. Additionally, the experiment will introduce a secondary parameter, $p2$, to satisfy the conditions of the Markov map. From p and $p2$, $p1$ will be derived using the specified formula.

$$p_1 = \frac{p}{1-p}p_2 \quad \begin{cases} p_2\text{: another parameter} \\ p_2 \neq t = 1-p \end{cases}$$

The theoretical values will be calculated based on the provided mathematical formula.

$$P_u = \frac{p_2}{p_1+p_2}\left((1-p_1)p_1(1-p_2) + p_1(1-p_2)p_2 + p_1p_2p_1\right) + \frac{p_1}{p_1+p_2}(p_2(1-p_1)p_1 + p_2p_1p_2 + (1-p_2)p_2(1-p_1) + (1-p_2)(1-p_2)(1-p_2))$$

Table 2. Probability of Undetected Error on Markov-type Map

p	$p1$	$p2$	Computed	Theoretical
0.05	0.008	0.16	0.04439	0.04413
0.05	0.018	0.34	0.04080	0.04073
0.1	0.018	0.16	0.08773	0.08819
0.1	0.038	0.34	0.08218	0.08167
0.15	0.028	0.16	0.13225	0.13219
0.15	0.060	0.34	0.12303	0.12281
0.2	0.040	0.16	0.17652	0.17609
0.2	0.085	0.34	0.16393	0.16411
0.25	0.053	0.16	0.21943	0.21987
0.25	0.113	0.34	0.20547	0.20552
0.3	0.069	0.16	0.26265	0.26349
0.3	0.146	0.34	0.24650	0.24696
0.35	0.086	0.16	0.30690	0.30690
0.35	0.183	0.34	0.28786	0.28826
0.4	0.107	0.16	0.35111	0.35002
0.4	0.227	0.34	0.32959	0.32915
0.45	0.131	0.16	0.39217	0.39273
0.45	0.278	0.34	0.36900	0.36915

0.49999	0.160	0.16	0.43469	0.43484
0.49999	0.340	0.34	0.40727	0.40744

Table 2 displays both the computed and theoretical values of the probability of undetected error, analyzing how these values change with incremental increases in the parameter p and the combined effects of $p1$ and $p2$. Similar to the observations in Table 1, the discrepancy between the computed and theoretical values is not substantial. As the parameter p increases, the probability of undetected error also rises. However, when the increments incorporate **a larger $p2$, there is a slight decrease in the probability of undetected errors** in both computed and theoretical assessments.

Appendix

1. Threshold function

$$\theta_t(x) = \begin{cases} 0 & (x < t) \\ 1 & (x \geq t) \end{cases}$$

2. Bernoulli map function

$$\tau(x, c) = \begin{cases} \dfrac{x}{c} & 0 \leq x < c \\ \dfrac{x - c}{1 - c} & c \leq x \leq 1 \end{cases}$$

3. Markov source parameters construction and function

Chaotic Map for Markov Source (Summary)

➢ Choose p_1, p_2 $(p_1 + p_2 \neq 1)$

$$t = \frac{p_2}{p_1 + p_2}, \quad a = \frac{1}{1 - (p_1 + p_2)} \begin{cases} > 0 & (p_1 + p_2 < 1) \\ < 0 & (p_1 + p_2 > 1) \end{cases}$$

	$a > 0$	$a < 0$
c_1	$t(1 - a^{-1})$	$t + (1 - t)a^{-1}$
c_2	$t + (1 - t)\, a^{-1}$	$t(1 - a^{-1})$
a_1	$\dfrac{1}{c_1}$	$\dfrac{1}{c_1}$
a_2	$\dfrac{1}{1 - c_2}$	$\dfrac{1}{1 - c_2}$
$\tau(x)$	$\begin{cases} a_1 x & (0 \leq x < c_1) \\ a(x - c_1) & (c_1 \leq x < c_2) \\ a_2(x - c_2) & (c_2 \leq x \leq 1) \end{cases}$	$\begin{cases} a_1 x & (0 \leq x < c_1) \\ a(x - c_2) & (c_1 \leq x < c_2) \\ a_2(x - c_2) & (c_2 \leq x \leq 1) \end{cases}$

-13-

4. Source code

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

def threshold_function(x, t):  # threshold function for making 0 and 1 value
    return 0 if x < t else 1
```

```
def skew_bernouli_map(x, c): # Bernoulli mapping function
    if x < c:
        return (x / c)
    else:
        return (x - c) / (1 - c)

def plm3(x, p_1, p_2, t): # Markov plm3 mapping function
    def create_parameters(p_1, p_2):
        a = 1 / (1 - (p_1 + p_2))
        if a > 0:
            a_positive = True
            c1 = t * (1 - (1/a))
            c2 = t + ((1 - t)/a)
            a1 = 1 / c1
            a2 = 1 / (1 - c2)
        else:
            a_positive = False
            c1 = t + ((1 - t)/a)
            c2 = t * (1 - (1/a))
            a1 = 1 / c1
            a2 = 1 / (1 - c2)

        return a, a_positive, c1, c2, a1, a2

    a, a_positive, c1, c2, a1, a2 = create_parameters(p_1, p_2)

    if x < c1:
        return a1 * x
    elif c1 <= x < c2:
        if a_positive:
            return a * (x - c1)
        else:
            return a * (x - c2)
    elif c2 <= x <= 1:
        return a2 * (x - c2)
    else:
        return None  # Handle case when x is outside [0, 1]

def memoryless_bernoulli():
    l = 1000000 # length (N)
    c = t = 0.49999 # t = c = 0.5 (~ 4.9999)
    def cte(p): # for sequence of errors with memoryless source (c=t=1-p)
        return 1 - p
    p_list = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.49999]

    for p in p_list:
        x0 = 0.1782612
        z0 = 0.5673244
```

```
        derr = 0 # counter for error

        for i in range (l):
            # information source and coding : Bernoulli map (c = t)
            x1 = skew_bernouli_map(x0, c); x2  = skew_bernouli_map(x1, c)
            b0 = threshold_function(x0, t); b1 = threshold_function(x1, t); b2 =
threshold_function(x2, t)
            b3 = b0 ^ b1 ^ b2
            x0 = skew_bernouli_map(x2, c) # prepare x0 for next loop

            # generating a sequence of errors with memoryless source (c=t=1-p) and information
xor error
            z1 = skew_bernouli_map(z0, cte(p)); z2 = skew_bernouli_map(z1, cte(p)); z3 =
skew_bernouli_map(z2, cte(p))
            e0 = threshold_function(z0, cte(p)); e1 = threshold_function(z1, cte(p)); e2 =
threshold_function(z2, cte(p)); e3 = threshold_function(z3, cte(p)) # error sequence
            r0 = b0 ^ e0; r1 = b1 ^ e1; r2 = b2 ^ e2; r3 = b3 ^ e3; # received sequence
            if (e0 == 1)|(e1 == 1)|(e2 == 1)|(e3 == 1):
                er = 1
            else:
                er = 0
            z0 = skew_bernouli_map(z3, cte(p)) # prepare z0 for next loop

            # error-detecting and count the number of undetected errors
            derr = derr + (r0 ^ r1 ^ r2 ^ r3 ^ er)

        # probability of undetected errors
        computed_value = derr / l
        theoretical_value = 6 * p**2 * (1-p)**2 + p**4

        # print the result
        print(f'For p: {p}, computed value: {computed_value:.5f} and theoretical value:
{theoretical_value:.5f}')

def markov():
    l = 1000000 # length (N)
    c = t = 0.49999 # t = c = 0.5 (~ 4.9999)
    def cte(p): # for sequence of errors with memoryless source (c=t=1-p)
        return 1 - p
    p_list = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.49999]
    p2_list = [0.16, 0.34] # another parameter p2

    for p in p_list:
        for p2 in p2_list:
            p1 = p / (1 - p) * p2
            x0 = 0.1782612
            z0 = 0.5673244
            derr = 0 # counter for error
```

```
            for i in range (l):
                # information source and coding : Bernoulli map (c = t)
                x1 = skew_bernouli_map(x0, c); x2  = skew_bernouli_map(x1, c)
                b0 = threshold_function(x0, t); b1 = threshold_function(x1, t); b2 =
threshold_function(x2, t)
                b3 = b0 ^ b1 ^ b2
                x0 = skew_bernouli_map(x2, c) # prepare x0 for next loop

                # generating a sequence of errors with markov-type source (c=t=1-p) and
information xor error
                z1 = plm3(z0, p1, p2, cte(p)); z2 = plm3(z1, p1, p2, cte(p)); z3 = plm3(z2, p1,
p2, cte(p))
                e0 = threshold_function(z0, cte(p)); e1 = threshold_function(z1, cte(p)); e2 =
threshold_function(z2, cte(p)); e3 = threshold_function(z3, cte(p)) # error sequence
                r0 = b0 ^ e0; r1 = b1 ^ e1; r2 = b2 ^ e2; r3 = b3 ^ e3; # received sequence
                if (e0 == 1)|(e1 == 1)|(e2 == 1)|(e3 == 1):
                    er = 1
                else:
                    er = 0
                z0 = plm3(z3, p1, p2, cte(p)) # prepare z0 for next loop

                # error-detecting and count the number of undetected errors
                derr = derr + (r0 ^ r1 ^ r2 ^ r3 ^ er)

            # probability of undetected errors
            computed_value = derr / l
            theoretical_value = (p2 / (p1 + p2)) * ((1 - p1) * p1 * (1 - p2) + p1 * (1 - p2) *
p2 + p1 * p2 * p1) + (p1 / (p1 + p2)) * (p2 * (1 - p1) * p1 + p2 * p1 * p2 + (1 - p2) * p2 * (1
- p1) + (1 - p2) * (1 - p2) * (1 - p2))

            # print the result
            print(f'For p: {p}, p1: {p1:.3f}, p2: {p2}; computed value: {computed_value:.5f}
and theoretical value: {theoretical_value:.5f}')

if __name__ == '__main__':
    print('\nMemoryless with bernoulli map\n')
    memoryless_bernoulli()
    print('\nMarkov\n')
    markov()
```