Fathan Ananta Nur
236-D8749

44135 Applied Information Theory
Assignment 3

In these experiments, the parameters below will be used:

```
Length (N) = 1000000
Initial value = 0.51262323
Parameter p1, p2 = {(0.01, 0.1), (0.4, 0.2), (0.9, 0.3), (0.9, 0.9)}
```

1. Design Markov information binary sequences based on the piecewise linear chaotic maps (PLM3).

   In this experiment, the Markov information binary was plot based on piecewise linear chaotic maps. The figures below illustrate the plots for different pairs of parameters, $p1$ and $p2$. Different conditions were utilized when $p1 + p2 < 1$ and $p1 + p2 > 1$.

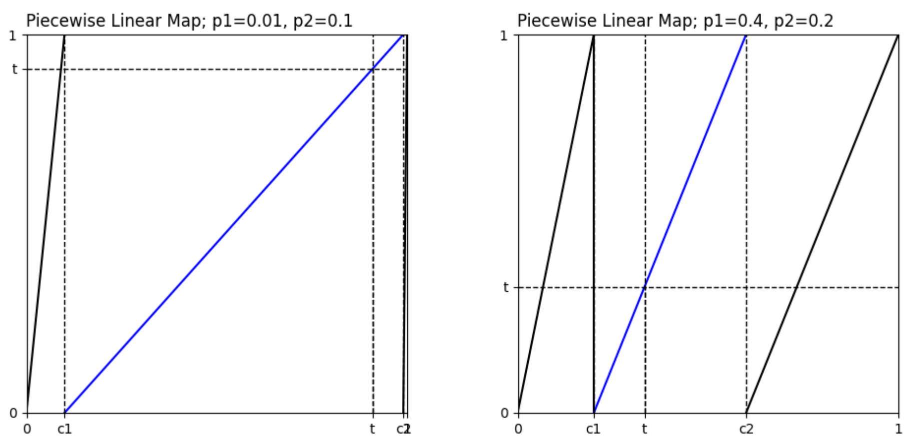

Figure 1. Piecewise Linear Chaotic Maps on $p1 + p2 < 1$ Condition

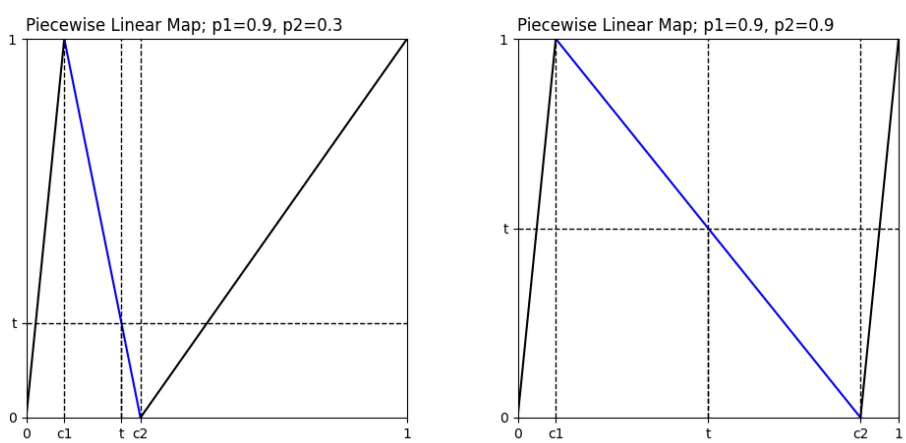

Figure 2. Piecewise Linear Chaotic Maps on $p1 + p2 > 1$ Condition

   As shown in Figure 1 and Figure 2, different conditions cause the slope a to skew in different directions. Parameters $p1$ and $p2$ significantly influence whether the slope a becomes positive or negative. When $p1 + p2 < 1$, the slope a becomes positive and **skews to the right**. Conversely, if $p1 + p2 > 1$, the slope will be negative and **skew to the left**. Condition when $p1 + p2 = 1$ cannot be done because the denominator becomes zero, and the calculation will be in error when divided by zero.

1.1. Confirm the chaotic maps have a uniform invariant density.

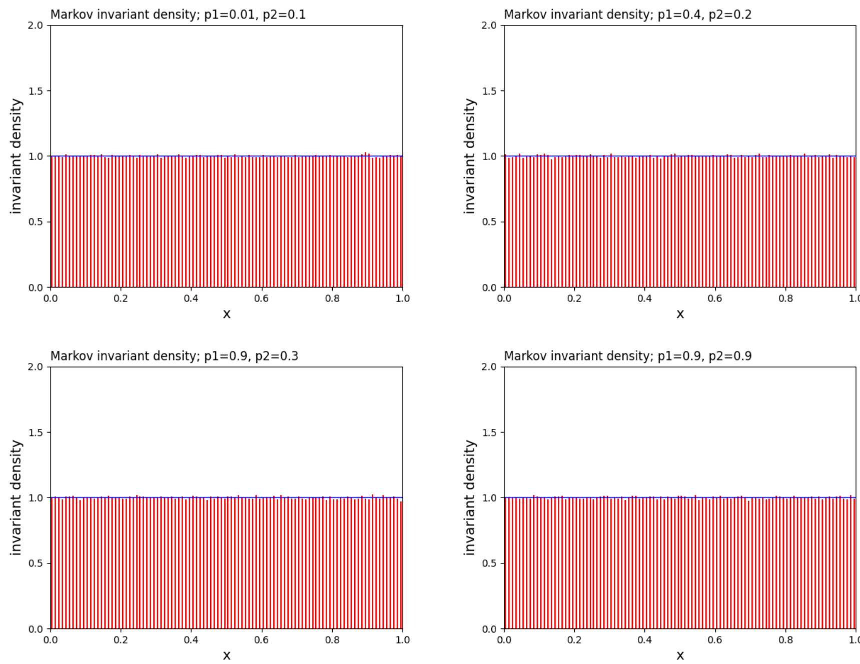

Figure 3. Invariant Density of PLM3 on Different Condition

Figure 3 demonstrates that altering the conditions of $p1$ and $p2$ does not affect the distribution, indicating the presence of an invariant distribution and proving **the chaotic maps have a uniform invariant density**. This invariance occurs because the system reaches a steady-state behavior, where the probability distribution of states remains unchanged despite variations in the parameters $p1$ and $p2$.

1.2. Generate binary sequences based on the designed maps and compute the following probabilities,

$$\mathrm{P}(0), \mathrm{P}(1), \mathrm{P}(00), \mathrm{P}(01), \mathrm{P}(10), \mathrm{P}(11)$$
$$\mathrm{P}(\mathrm{S}_0|\mathrm{S}_0) = \frac{\mathrm{P}(00)}{\mathrm{P}(0)}, \mathrm{P}(\mathrm{S}_1|\mathrm{S}_0) = \frac{\mathrm{P}(01)}{\mathrm{P}(0)},$$
$$\mathrm{P}(\mathrm{S}_0|\mathrm{S}_1) = \frac{\mathrm{P}(10)}{\mathrm{P}(1)}, \mathrm{P}(\mathrm{S}_1|\mathrm{S}_1) = \frac{\mathrm{P}(11)}{\mathrm{P}(1)}.$$

In the experiment, various values were used for the parameters $p1$ and $p2$, including both cases where $p1 + p2 < 1$ and $p1 + p2 > 1$. The computed values obtained directly from the program were compared with the theoretical values derived using the theoretical formula based on $p1$ and $p2$. This comparison is crucial because it allows to verify whether the value of computational model will be nearly same as the theoretical predictions, proving the memoryless source.

Table 1. Comparison of Computed Value and Theoretical Value When $p1 = 0.01$ and $p2 = 0.1$

	Computed value (from code calculation)	Theoretical value	Theoretical value formula
$\mathrm{P}(0)$	0.90953	0.909	$\frac{p2}{p1+p2}$
$\mathrm{P}(1)$	0.09047	0.091	$\frac{p1}{p1+p2}$
$\mathrm{P}(00)$	0.90046	0.9	$\frac{p2(1-p1)}{p1+p2}$
$\mathrm{P}(01)$	0.00907	0.009	$\frac{p2(p1)}{p1+p2}$
$\mathrm{P}(10)$	0.00907	0.009	$\frac{p1(p2)}{p1+p2}$
$\mathrm{P}(11)$	0.08141	0.081	$\frac{p1(1-p2)}{p1+p2}$
$\mathrm{P}(\mathrm{S}_0\|\mathrm{S}_0)$	0.99003	0.99	$1-p1$
$\mathrm{P}(\mathrm{S}_0\|\mathrm{S}_1)$	0.10022	0.1	$p2$
$\mathrm{P}(\mathrm{S}_1\|\mathrm{S}_0)$	0.00997	0.01	$p1$
$\mathrm{P}(\mathrm{S}_1\|\mathrm{S}_1)$	0.89978	0.9	$1-p2$

Table 2. Comparison of Computed Value and Theoretical Value When $p1 = 0.4$ and $p2 = 0.2$

	Computed value (from code calculation)	Theoretical value	Theoretical value formula
$\mathrm{P}(0)$	0.33326	0.333	$\frac{p2}{p1+p2}$
$\mathrm{P}(1)$	0.66674	0.667	$\frac{p1}{p1+p2}$
$\mathrm{P}(00)$	0.20005	0.2	$\frac{p2(1-p1)}{p1+p2}$
$\mathrm{P}(01)$	0.13321	0.133	$\frac{p2(p1)}{p1+p2}$
$\mathrm{P}(10)$	0.13321	0.133	$\frac{p1(p2)}{p1+p2}$
$\mathrm{P}(11)$	0.53353	0.533	$\frac{p1(1-p2)}{p1+p2}$
$\mathrm{P}(\mathrm{S}_0\|\mathrm{S}_0)$	0.60029	0.6	$1-p1$
$\mathrm{P}(\mathrm{S}_0\|\mathrm{S}_1)$	0.19979	0.2	$p2$
$\mathrm{P}(\mathrm{S}_1\|\mathrm{S}_0)$	0.39971	0.4	$p1$
$\mathrm{P}(\mathrm{S}_1\|\mathrm{S}_1)$	0.80021	0.8	$1-p2$

Table 3. Comparison of Computed Value and Theoretical Value When $p1 = 0.9$ and $p2 = 0.3$

	Computed value (from code calculation)	Theoretical value	Theoretical value formula

P(0)	0.25035	0.25	$\frac{p2}{p1+p2}$
P(1)	0.74965	0.75	$\frac{p1}{p1+p2}$
P(00)	0.02516	0.025	$\frac{p2(1-p1)}{p1+p2}$
P(01)	0.22519	0.225	$\frac{p2(p1)}{p1+p2}$
P(10)	0.22519	0.225	$\frac{p1(p2)}{p1+p2}$
P(11)	0.52445	0.525	$\frac{p1(1-p2)}{p1+p2}$
$P(S_0 \mid S_0)$	0.10049	0.1	$1-p1$
$P(S_0 \mid S_1)$	0.30040	0.3	$p2$
$P(S_1 \mid S_0)$	0.89951	0.9	$p1$
$P(S_1 \mid S_1)$	0.69960	0.7	$1-p2$

Table 4. Comparison of Computed Value and Theoretical Value When $p1 = 0.9$ and $p2 = 0.9$

	Computed value (from code calculation)	Theoretical value	Theoretical value formula
P(0)	0.49997	0.5	$\frac{p2}{p1+p2}$
P(1)	0.50003	0.5	$\frac{p1}{p1+p2}$
P(00)	0.04990	0.05	$\frac{p2(1-p1)}{p1+p2}$
P(01)	0.45008	0.45	$\frac{p2(p1)}{p1+p2}$
P(10)	0.45008	0.45	$\frac{p1(p2)}{p1+p2}$
P(11)	0.04995	0.05	$\frac{p1(1-p2)}{p1+p2}$
$P(S_0 \mid S_0)$	0.09980	0.1	$1-p1$
$P(S_0 \mid S_1)$	0.90011	0.9	$p2$
$P(S_1 \mid S_0)$	0.90020	0.9	$p1$
$P(S_1 \mid S_1)$	0.09989	0.1	$1-p2$

Table 1, Table 2, Table 3, and Table 4 demonstrate that for various values of $p1$ and $p2$, the computed values from the program closely match the theoretical values derived from the formula. This consistency **indicates the memoryless source** on the program. In a memoryless source, the probability of transitioning from one state to another depends only on the current state and not on the previous states. This property simplifies the theoretical calculations and ensures that the computed probabilities are straightforward to verify. Additionally, the accuracy of the program in replicating these theoretical probabilities suggests that the implementation correctly captures the underlying stochastic processes governed by $p1$ and $p2$.

2. Generate a Markov information binary sequence based on a PLM3 map and save it as a text file. Next compress the file with a file compression software. By performing this for several values of $p1$ and $p2$, consider the relation between the entropy and the compression ratio.

   In the following experiment, parameters $p1$ and $p2$ will be employed to observe the behavior of entropy. The conditions across $p1 + p2 < 1$ and $p1 + p2 > 1$ will be examined to understand how entropy behaves across these different parameter settings.

```
Length (N) = 1000000
Initial value = 0.51262323
Parameter p1, p2 = {(0.01, 0.01), (0.05, 0.05), (0.1, 0.2), (0.1, 0.3), (0.2,
0.4), (0.2, 0.5), (0.3, 0.5), (0.4, 0.4), (0.4999, 0.5), (0.6, 0.7), (0.6, 0.8),
(0.7, 0.8), (0.7, 0.9), (0.8, 0.9), (0.8, 0.95), (0.9, 0.95)}
```

$$H(p_1, p_2) = \frac{p_2}{p_1 + p_2}\left(-p_1 log_2 p_1 - (1 - p_1) log_2 (1 - p_1)\right) + \frac{p_1}{p_1 + p_2}\left(-p_2 log_2 p_2 - (1 - p_2) log_2 (1 - p_2)\right)$$

   For each pair of parameters $p1$ and $p2$, a binary sequence will be generated and saved as a text file (.txt). The text file is then compressed by ZIP, which is available as standard on the Windows 11 or macOS. Formula above is used to calculate the entropy value on the pair of parameters $p1$ and $p2$.

Table 5. Effect of Entropy on Compression Ratio

p_1	p_2	$H(p_1, p_2)$	Uncompressed size (byte)	Compressed size (byte)	Compression ratio
0.01	0.01	0.08079314	1000000	18502	54.048211
0.05	0.05	0.28639696	1000000	55092	18.1514557
0.1	0.2	0.55330643	1000000	104199	9.59702108
0.1	0.3	0.57206942	1000000	108643	9.20445864
0.2	0.4	0.80493559	1000000	142296	7.02760443
0.2	0.5	0.80137721	1000000	143237	6.98143636
0.3	0.5	0.92580681	1000000	156113	6.40561644
0.4	0.4	0.97095059	1000000	158445	6.31133832
0.4999	**0.5**	**0.99999999**	**1000000**	**159315**	**6.27687286**
0.6	0.7	0.9295692	1000000	155417	6.43430255
0.6	0.8	0.86422667	1000000	148033	6.75525052
0.7	0.8	0.80692159	1000000	140833	7.10060852
0.7	0.9	0.7009117	1000000	125481	7.969334
0.8	0.9	0.60290104	1000000	110211	9.07350446
0.8	0.95	0.52282815	1000000	97371	10.2699983
0.9	0.95	0.38016382	1000000	71681	13.9506982

Based on the findings from Table 5, as the parameters $p1$ and $p2$ were varied, the entropy value converges to 1 when the sum $p1 + p2$ approaches 1 from either direction. This convergence suggests that **when $\boldsymbol{p1 + p2}$ nears 1, there is greater variability** in the binary sequence generated. Furthermore, Table 5 allowed a graph illustrating the relationship between the entropy value and the compression ratio to be plotted. This graph provides insights into how changes in entropy correspond to variations in the compression ratio value across different parameter settings $p1$ and $p2$.

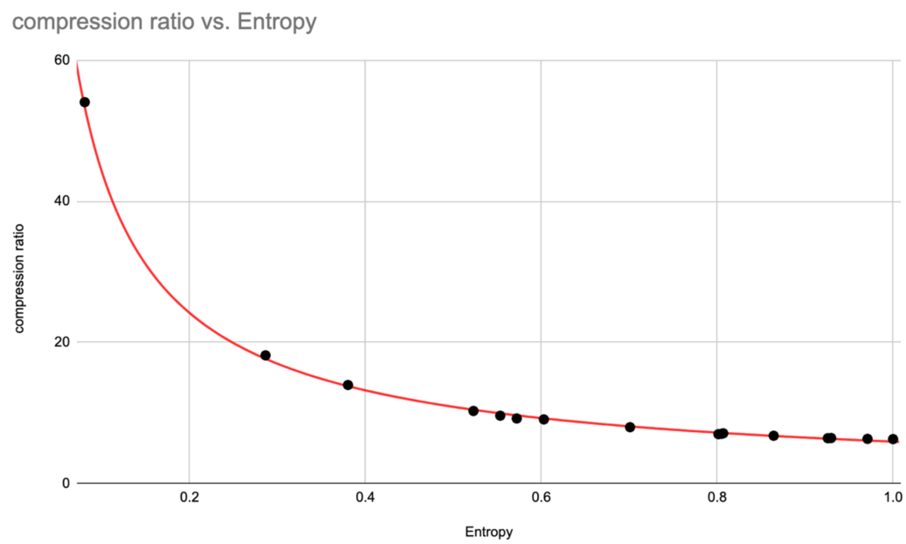

Figure 4. Relationship Between Entropy and Compression Ratio

Figure 4 illustrates that the value of entropy gradually increases as the value of parameters $p1$ and $p2$ varies fulfilling conditions across $p1 + p2 < 1$ and $p1 + p2 > 1$. The results show that the compression ratio gradually decreases as the entropy increases. These observations suggest that **higher entropy indicates a greater variability** in the bit sequence. Consequently, sequences with greater variability result in larger compression ratios compared to those with lower entropy. As can be shown also on the previous assignment (Assignment 2), when the entropy value approaches 1, the compression ratio consistently converges to 6.

Appendix

1. Formulas used

Chaotic Map for Markov Source (Summary)

- Choose $p_1, p_2 \quad (p_1 + p_2 \neq 1)$

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

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

-13-

2. Source code

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

def create_parameters(p_1, p_2):
    t = p_2 / (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 t, a, a_positive, c1, c2, a1, a2

def threshold_function(x, t):
    return 0 if x < t else 1

def plm3(a_positive, x, a, a1, a2, c1, c2):
    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 generate_sequence(x0, a_positive, a, a1, a2, c1, c2, l): # generate sequence using plm3 map
    x = np.zeros(l)
    x[0] = x0
    for i in range(1, l):
        x[i] = plm3(a_positive, x[i-1], a, a1, a2, c1, c2)
    return x

def plot(p_1, p_2, t, a_positive, a, a1, a2, c1, c2, x0, l):
    x = np.arange(0, 1.00000, 0.00001)
    y = np.array([plm3(a_positive, xi, a, a1, a2, c1, c2) for xi in x])

    # piecewise linear chaotic map 3
    x_ticks = [0, c1, t, c2, 1]
    x_labels = ['0', 'c1', 't', 'c2', '1']
    y_ticks = [0, t, 1]
    y_labels = ['0', 't', '1']

    c1_range = int(c1 / 0.00001) + 1
    c2_range = int(c2 / 0.00001) + 1

    plt.figure(figsize=(5, 5))
    plt.plot(x[:c1_range], y[:c1_range], color='k')
    plt.plot(x[c1_range:c2_range], y[c1_range:c2_range], color='b')
    plt.plot(x[c2_range:], y[c2_range:], color='k')
    plt.xlim(0, 1)
    plt.ylim(0, 1)
    plt.xticks(x_ticks, x_labels)
    plt.yticks(y_ticks, y_labels)
    plt.vlines(t, 0, 1, color='k', linewidth=1, linestyles='dashed')
```

```
    plt.vlines(c2, 0, 1, color='k', linewidth=1, linestyles='dashed')
    plt.vlines(c1, 0, 1, color='k', linewidth=1, linestyles='dashed')
    plt.vlines(t, 0, t, color='k', linewidth=1, linestyles='dashed')
    plt.hlines(t, 0, 1, color='k', linewidth=1, linestyles='dashed')
    plt.title(f'Piecewise Linear Map; p1={p_1}, p2={p_2}', loc="left")
    plt.savefig(f"assignment3/1/MarkovMap_p1:{p_1}_p2:{p_2}.png")

    # generate sequence
    sequence = generate_sequence(x0, a_positive, a, a1, a2, c1, c2, l)

    # invariant density
    plt.figure()
    plt.hist(sequence, bins=100, rwidth=0.4, color='r', density=True)
    plt.xlim(0, 1)
    plt.ylim(0, 2)
    plt.yticks([0, 0.5, 1, 1.5, 2])
    plt.hlines(1, 0, 1, color='b', linewidth=1)
    plt.xlabel("x", fontsize=14)
    plt.ylabel("invariant density", fontsize=14)
    plt.title(f'Markov invariant density; p1={p_1}, p2={p_2}', loc="left")
    plt.savefig(f"assignment3/1/density_p1:{p_1}_p2:{p_2}.png")

def main():
    p_list = [(0.01, 0.1), (0.4, 0.2), (0.9, 0.3), (0.9, 0.9)]  # p list
    x0 = 0.51262323  # initial value
    l = 1000000  # length (N)

    for p in p_list:
        p_1, p_2 = p
        t, a, a_positive, c1, c2, a1, a2 = create_parameters(p_1, p_2)
        os.makedirs('assignment3/1', exist_ok=True)
        plot(p_1, p_2, t, a_positive, a, a1, a2, c1, c2, x0, l)

        c1_count = c00 = c01 = c10 = c11 = 0  # initialization of counters
        for i in range(l):
            b1 = threshold_function(x0, t)
            next_x = plm3(a_positive, x0, a, a1, a2, c1, c2)
            if next_x is None:  # Ensure valid value --> refer the last condition on plm3 function
                continue
            b2 = threshold_function(next_x, t)
            c1_count += b1  # number of 1
            c11 += b1 * b2
            c10 += b1 * (1 - b2)
            c01 += (1 - b1) * b2
            c00 += (1 - b1) * (1 - b2)
            x0 = next_x  # next mapping

        # calculate P
```

```
        p1 = c1_count / l
        p0 = 1 - p1
        p00 = c00 / l
        p01 = c01 / l
        p10 = c10 / l
        p11 = c11 / l
        p0_0 = p00 / p0  # P(S0|S0)
        p0_1 = p10 / p1  # P(S0|S1)
        p1_0 = p01 / p0  # P(S1|S0)
        p1_1 = p11 / p1  # P(S1|S1)

        # display - 5 values behind comma
        print(f'parameter p1: {p_1}, p2: {p_2} --> t: {t:.3f}, a: {a:.3f}, c1: {c1:.3f}, c2:
{c2:.3f}, a1: {a1:.3f}, a2: {a2:.3f}')
        print(f'P(0): {p0:.5f}')
        print(f'P(1): {p1:.5f}')
        print(f'P(00): {p00:.5f}')
        print(f'P(01): {p01:.5f}')
        print(f'P(10): {p10:.5f}')
        print(f'P(11): {p11:.5f}')
        print(f'P(0|0): {p0_0:.5f}')
        print(f'P(0|1): {p0_1:.5f}')
        print(f'P(1|0): {p1_0:.5f}')
        print(f'P(1|1): {p1_1:.5f}')

def main2():
    p_list = [(0.01, 0.01), (0.05, 0.05), (0.1, 0.2), (0.1, 0.3), (0.2, 0.4), (0.2, 0.5), (0.3,
0.5), (0.4, 0.4), (0.4999, 0.5), (0.6, 0.7), (0.6, 0.8), (0.7, 0.8), (0.7, 0.9), (0.8, 0.9), (0.8,
0.95), (0.9, 0.95)]  # p list
    x0 = 0.51262323   # initial value
    l = 1000000  # length (N)

    for p in p_list:
        p_1, p_2 = p
        t, a, a_positive, c1, c2, a1, a2 = create_parameters(p_1, p_2)
        b_seq = ""
        for i in range(l):
            b1 = threshold_function(x0, t)
            b_seq += str(b1)
            x0 = plm3(a_positive, x0, a, a1, a2, c1, c2)

        # check result
        print(f"p1:{p_1}, p2:{p_2}", b_seq[:10])
        print("length", len(b_seq))
        # save
        os.makedirs(f'assignment3/2/p1:{p_1}, p2:{p_2}', exist_ok=True)
        with open(f'assignment3/2/p1:{p_1}, p2:{p_2}/p1:{p_1}, p2:{p_2}.txt', 'w') as f:
            f.write(b_seq)
```

```python
if __name__ == "__main__":
    print('\n\nnumber 1\n')
    main()
    print('\n\nnumber 2\n')
    main2()
```