19CCE303 – DIGITAL COMMUNICATION

ASSIGNMENT III

PROBLEM:

Fundamentals of Communication Systems, John G. Proakis, Masoud Salehi, Pearson Education, Second Edition, 2015

Chapter 10 – Digital Transmission through Bandlimited Channels

Computer Problem 10.6 Simulation of Detector Performance for Duobinary Signal

The objective of this problem is to simulate a binary PAM communication system that employs a duobinary signal pulse, where the precoding and amplitude conversion that yields the sequence $\{a_n\}$ are performed as prescribed in the text. Hence, the input to the detector is the sequence

$$y_k = b_k + n_k$$
$$= a_k + a_k\text{-}1 + n_k, \qquad k + 1, 2,$$

where the sequence $\{n_k\}$ is zero mean, Gaussian, and uncorrelated. The variance of n_k is σ^2.

Perform the simulation for 10,000 bits and measure the bit-error probability for $\sigma^2 = 0.1$, $\sigma^2 = 0.5$, and $\sigma^2 = 1$. Plot the theoretical error probability for binary PAM with no ISI, and compare the simulation results with this ideal performance. You should observe some degradation in the performance of the duobinary system. What are the approximate values of the degradation for $\sigma^2 = 0.1$, $\sigma^2 = 0.5$, and $\sigma^2 = 1$?

SOFTWARE REQUIRED:

- Windows 10 Operating System
- Anaconda3 2021.11 (Python 3.9.7 64-bit)
- The Scientific Python Development Environment (Spyder) 5.1.5

PYTHON CODE:

```
import numpy as np # Support for large, multi-dimensional arrays and matrices

x = np.random.uniform(0.0, 1.0, 10001) # Draw samples from a uniform
distribution
binary_data = []
for i in x:
    if i>0.5:
        binary_data.append(1)
    elif i<=0.5:
        binary_data.append(0)
print("\nFirst 100 Binary Data Sequence, {dk}: ", binary_data[0:100])
```

```
# Precodes it for a duobinary pulse transmission system to produce the
sequence, {pk}: -
pn =[]; pn.append(0); m = 2
for i in range(len(binary_data)):
    pn.append((binary_data[i] - pn[i]) % m)
print("\nFirst 100 Precoded and Produced Sequence, {pk}: ", pn[0:100])

# Maps the precoded sequence into the transmitted amplitude levels, {ak}: -
an = []
for i in range(0,len(pn)):
    an.append(2*pn[i] - (m-1))
print("\nFirst 100 Transmitted Amplitude Levels, {ak}: ", an[0:100])

# Received noise-free sequences, {bk}: -
bn = []
for i in range(1,len(pn)):
    bn.append(an[i] + an[i-1])
print("\nFirst 100 Received Noise-Free Sequences, {bk}: ", bn[0:100])

# Recover the original data sequence: -
dn = []
for i in range(len(bn)):
    dn.append(int((((bn[i]/2) + (m-1)) % m)))
print("\nFirst 100 Recovered Original Data Sequence: ", dn[0:100])

print("\nWith Noise (Draw random samples from a normal (Gaussian)
distribution): -")

pn1 = []; pn1.append(0)
pn2 = []; pn2.append(0)
pn3 = []; pn3.append(0)

for i in range(len(binary_data)):
    pn1.append((binary_data[i] - pn1[i]) % m)
    pn2.append((binary_data[i] - pn2[i]) % m)
    pn3.append((binary_data[i] - pn3[i]) % m)

print("\nFirst 10 Precoded and Produced Sequence With Noise, {pk}: ")
print("pn1: ", pn1[0:10])
print("pn2: ", pn2[0:10])
print("pn3: ", pn3[0:10])

an1 = an2 = an3 = []
for i in range(len(pn1)):
    an1.append(2*pn1[i] - (m-1))
    an2.append(2*pn2[i] - (m-1))
    an3.append(2*pn3[i] - (m-1))
```

```
print("\nFirst 10 Transmitted Amplitude Levels With Noise, {ak}: ")
print("an1: ", an1[0:10])
print("an2: ", an2[0:10])
print("an3: ", an3[0:10])

n1 = np.random.normal(0,0.1,30006)
n2 = np.random.normal(0,0.5,30006)
n3 = np.random.normal(0,1,30006)

noise1 = noise2 = noise3 = []
for i in range(len(an1)):
    noise1.append(an1[i]+n1[i])
    noise2.append(an2[i]+n2[i])
    noise3.append(an3[i]+n3[i])

bn1 = bn2 = bn3 = []
for i in range(1, len(noise1)):
    bn1.append(noise1[i]+noise1[i-1])
    bn2.append(noise2[i]+noise2[i-1])
    bn3.append(noise3[i]+noise3[i-1])

print("\nFirst 10 Received Noise Sequences, {bk}: ")
print("bn1: ", bn1[0:10])
print("bn2: ", bn2[0:10])
print("bn3: ", bn3[0:10])

dn1 = dn2 = dn3 = []
for i in range(len(bn1)):
    dn1.append((((bn1[i]/2) + (m-1)) % m))
    dn2.append((((bn2[i]/2) + (m-1)) % m))
    dn3.append((((bn3[i]/2) + (m-1)) % m))

print("\nFirst 10 Recovered Original Data Sequence Before Threshold: ")
print("dn1: ", dn1[0:10])
print("dn2: ", dn2[0:10])
print("dn3: ", dn3[0:10])

final1 = final2 = final3 = []
for i in range(len(bn1)):
    if dn1[i] > 0.5 or dn1[i] < -0.5:
        final1.append(0)
    else:
        final1.append(1)
    if dn2[i] > 0.5 or dn2[i] < -0.5:
        final2.append(0)
    else:
        final2.append(1)
```

```
    if dn3[i] > 0.5 or dn3[i] < -0.5:
        final3.append(0)
    else:
        final3.append(1)

print("\nFirst 10 Recovered Original Data Sequence After Threshold: ")
print("Final1: ", final1[0:10])
print("Final2: ", final2[0:10])
print("Final3: ", final3[0:10])

error_count1 = error_count2 = error_count3 = 0
for i in range(10001):
    if binary_data[i] != final1[i]:
        error_count1 += 1
    if binary_data[i] != final2[i]:
        error_count2 += 1
    if binary_data[i] != final3[i]:
        error_count3 += 1
print("\nError in Variance of 0.1: ", error_count1)
print("Error in Variance of 0.5: ", error_count2)
print("Error in Variance of 1: ", error_count3)
```

INFERENCE:

Inter Symbol Interference (ISI) causes degradation in the performance of the system.

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

Restarting kernel...
```

```
In [1]:         'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester V/
19CCE303 - Digital Communication/Assignments/Assignment III/
19CCE303_Assignment_III_Code.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject
Materials/Semester V/19CCE303 - Digital Communication/Assignments/Assignment III'

First 100 Binary Data Sequence, {dk}:  [1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1,
0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]

First 100 Precoded and Produced Sequence, {pk}:  [0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0]

First 100 Transmitted Amplitude Levels, {ak}:  [-1, 1, 1, 1, 1, 1, -1, 1, 1, -1, 1, 1, 1,
1, 1, 1, 1, 1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, 1, -1, -1, -1,
-1, -1, -1, 1, -1, 1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, 1, -1, 1, -1, 1, -1, -1, 1, -1,
1, -1, 1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, 1, -1, -1, 1, -1, 1, 1, -1, -1, -1, 1,
-1, 1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1]

First 100 Received Noise-Free Sequences, {bk}:  [0, 2, 2, 2, 2, 0, 0, 2, 0, 0, 2, 2, 2, 2,
2, 2, 2, 0, 0, 0, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, 0, 0, 0, 0, -2, -2, -2, -2, -2,
0, 0, 0, 0, -2, -2, -2, 0, 2, 2, 0, -2, -2, 0, 0, 0, 0, 0, 0, -2, 0, 0, 0, 0, 0, 0, -2, 0,
2, 2, 2, 0, -2, -2, -2, -2, 0, 0, -2, 0, 0, 0, 2, 0, -2, -2, 0, 0, 0, 0, -2, -2, -2, 0, 2,
2, 2, 0, -2, -2, -2]

First 100 Recovered Original Data Sequence:  [1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]

With Noise (Draw random samples from a normal (Gaussian) distribution): -

First 10 Precoded and Produced Sequence With Noise, {pk}:
pn1:  [0, 1, 1, 1, 1, 1, 0, 1, 1, 0]
pn2:  [0, 1, 1, 1, 1, 1, 0, 1, 1, 0]
pn3:  [0, 1, 1, 1, 1, 1, 0, 1, 1, 0]

First 10 Transmitted Amplitude Levels With Noise, {ak}:
an1:  [-1, -1, -1, 1, 1, 1, 1, 1, 1, 1]
an2:  [-1, -1, -1, 1, 1, 1, 1, 1, 1, 1]
an3:  [-1, -1, -1, 1, 1, 1, 1, 1, 1, 1]

First 10 Received Noise Sequences, {bk}:
bn1:  [-2.498929105619313, -2.498929105619313, -2.498929105619313, -1.4239547509633006,
```

```
-1.4239547509633006, -1.4239547509633006, -0.9154554090691746, -0.9154554090691746,
-0.9154554090691746, -1.2967855426826294]
bn2:  [-2.498929105619313, -2.498929105619313, -2.498929105619313, -1.4239547509633006,
-1.4239547509633006, -1.4239547509633006, -0.9154554090691746, -0.9154554090691746,
-0.9154554090691746, -1.2967855426826294]
bn3:  [-2.498929105619313, -2.498929105619313, -2.498929105619313, -1.4239547509633006,
-1.4239547509633006, -1.4239547509633006, -0.9154554090691746, -0.9154554090691746,
-0.9154554090691746, -1.2967855426826294]

First 10 Recovered Original Data Sequence Before Threshold:
dn1:  [1.7505354471903436, 1.7505354471903436, 1.7505354471903436, 1.7505354471903436,
1.7505354471903436, 1.7505354471903436, 1.7505354471903436, 1.7505354471903436,
1.7505354471903436, 0.2880226245183497]
dn2:  [1.7505354471903436, 1.7505354471903436, 1.7505354471903436, 1.7505354471903436,
1.7505354471903436, 1.7505354471903436, 1.7505354471903436, 1.7505354471903436,
1.7505354471903436, 0.2880226245183497]
dn3:  [1.7505354471903436, 1.7505354471903436, 1.7505354471903436, 1.7505354471903436,
1.7505354471903436, 1.7505354471903436, 1.7505354471903436, 1.7505354471903436,
1.7505354471903436, 0.2880226245183497]

First 10 Recovered Original Data Sequence After Threshold:
Final1:  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Final2:  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Final3:  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Error in Variance of 0.1:  5024
Error in Variance of 0.5:  5024
Error in Variance of 1:  5024

In [2]:
```