Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Томский государственный университет систем управления и радиоэлектроники

(ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ОТЧЕТ

К лабораторной работе 2 по дисциплине “Сети и телекоммуникации”

Вариант - 3

Студенты гр. 589-3

Бураков И. С.

Потлог А.М.

Слесаренко А.С.

Хромов В.В.

21.09.2022

Принял:

Профессор каф. КСУП

Преподаватель

Томск 2022

Оглавление

[**1 Введение** 3](#_Toc115824113)

[**2 Задание** 4](#_Toc115824114)

[**3 Фазовая манипуляция** 5](#_Toc115824115)

[**4 Частотная манипуляция** 6](#_Toc115824116)

[**5 Построение фазового приемника** 7](#_Toc115824117)

[**6 Построение частотного приемника** 10](#_Toc115824118)

[**7 Заключение** 13](#_Toc115824119)

[**8 Ответы на контрольные вопросы** 14](#_Toc115824120)

[**9 Приложение А** 15](#_Toc115824121)

# **1 Введение**

Модуляция — преобразование первичного сообщения (сигнала) в сигнал, пригодный для передачи по линии связи.

Демодуляция — это не только операция, обратная модуляции, выполняемые над пришедшем из канала сигналом. В результате различных искажений и воздействия помех, пришедший сигнал может существенно отличаться от переданного сигнала. Поэтому демодуляция должна включать анализ приходящего сигнала, и принимать решение о переданном сообщении.

Дискретную модуляцию часто называют манипуляцией, и устройство его осуществляющую — манипулятор или генератор сигналов.

# **2 Задание**

1. Составить программу с частотной и фазовой манипуляцией;

2. Исследовать спектры полученных сигналов;

3. Принять манипулированные сигналы для частотной и фазовой

манипуляции;

4. Принять манипулированные сигналы с помехами для частотной и

фазовой манипуляции;

5. Сделать выводы.

6. Оформить отчет.

Вариант

Порог определения фазового приемника Pd, с – 0.39

Порог срабатывания частотного приемника Fd, с – 0.37

Входной сигнал - 01111101

# **3 Фазовая манипуляция**

Фазовая манипуляция применяется намного чаще чем фазовая модуляция, и проще для понимания. Простая фазовая манипуляция показана в примерах дальше. Исходный набор данных представляет собой последовательность из DataLng=8 бит, которые записаны в массив Data. При фазовой манипуляции используется модуль PhaseManipulation.py.

где Data — исходный массив данных

А = 1 — амплитуда модулированного колебания;

fsin — несущая частота колебаний;

С=Tсигн/K — интервал дискретизации;

N = 2^7 — количество точек несущей, на которые при передаче накладывается один бит данных.

Полученная фазовая манипуляция приведена на рис. 3.1. Спектр фазовой манипуляции приведён на рис. 3.2.

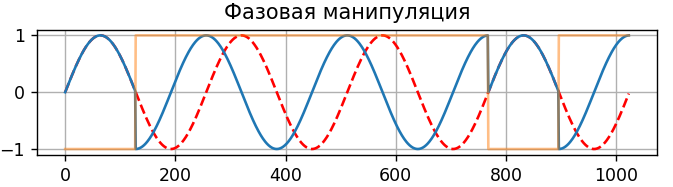


Рисунок 3.1 – Фазовая манипуляция

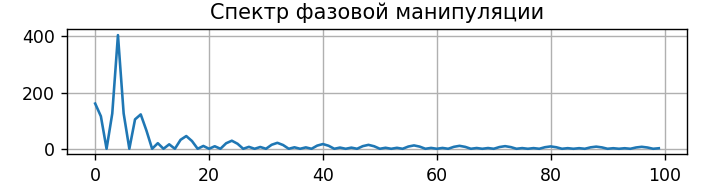


Рисунок 3.2 – Спектр фазовой манипуляции

# **4 Частотная манипуляция**

Набор начальных данных тот же что и в предыдущем примере. При моделировании частотной манипуляции используется модуль FrequencyModulation.py

где Data — исходный массив данных

А = 1 — амплитуда модулированного колебания;

fsin — несущая частота колебаний;

Wd — амплитуда частотного отклонения (частота девиации);

С=Tсигн/K — интервал дискретизации;

Tсигн=DataLng/fsin – длительность манипулированного сигнала;

N = 2^7 — количество точек несущей, на которые при передаче накладывается один бит данных.

Манипулированный сигнал представлен на рис. 4.3. На рис. 4.4

изображён спектр этого сигнала.



Рисунок 4.3 – Частотная манипуляция



Рисунок 4.4 – Спектр частотной манипуляции

# **5 Построение фазового приемника**

На вход приёмника поступает фазово-манипулированный сигнал.

Реализация этой формулы представлена в модуле PhaseDemodulator.py. Здесь L1— массив дискретов сигнала, соответствующего передаче «0», L2 — массив дискретов сигнала, соответствующего передаче «1». Результатами этих программ являются вектора sb1 и sb2, в которых записаны последовательности битов («1» соответствует ожидаемой частоте, «0» — в обратном случае). Таким образом, в векторе sb2, будет содержаться исходная последовательность битов, записанная в векторе Data.

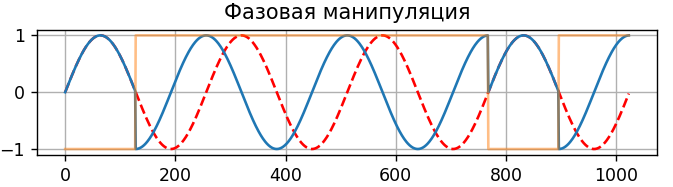


Рисунок 5.5 – Исходный манипулированный сигнал

sb1 = [[0, 105, 105, 105, 105, 105, 0, 105],

[0, 1, 1, 1, 1, 1, 0, 1]]

sb2 = [[105, 0, 0, 0, 0, 0, 105, 0],

[1, 0, 0, 0, 0, 0, 1, 0]]

Выходной сигнал изображён на рис. 5.6 и рис. 5.7.



Рисунок 5.6 – Демодулированный сигнал



Рисунок 5.6 – Демодулированный сигнал

Проверка сигнала на помехоустойчивость:

ss = [i + random.uniform(0, 0.5) - random.uniform(0, 0.5) for i in s]

sb1 = [[0, 105, 105, 105, 105, 105, 0, 105],

[0, 1, 1, 1, 1, 1, 0, 1]]

sb2 = [[105, 0, 0, 0, 0, 0, 105, 0],

[1, 0, 0, 0, 0, 0, 1, 0]]

Исходный сигнал с помехами показан на рис. 5.8, демодулированный сигнал на рис. 5.6 и рис. 5.7 (совпадают с принятыми без помех). Как можно заметить сигнал был принят без ошибок, но в некоторых битах был близок к этому.

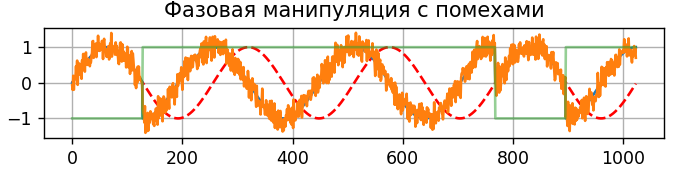


Рисунок 5.8 – Манипулированный исходный сигнал с помехами

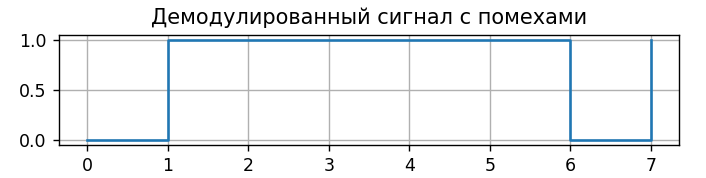


Рисунок 5.9 – Демодулированный сигнал с помехами



Рисунок 5.10 – Демодулированный сигнал с помехами

# **6 Построение частотного приемника**

На вход приёмника поступает частотно-манипулированный сигнал, для того бы подавить помехи, не прибегая к применению частотных фильтров, будем переводить принятый сигнал в импульсы, которые в дальнейшем будем считать. Для того чтобы перевести принятый сигнал в импульсы введём понятие состояние приёмника, которое может принимать 3 значения:

• «0» — приёмник в первоначальном состоянии

• «1» — приёмник принял сигнал больше порога с положительной

амплитудой

• «-1» — приёмник принял сигнал больше порога с отрицательной

амплитудой.

В начале каждого периода приёмник переводится в состояние «0», переход в любое другое состояние инициирует появление импульса на выходе. Легко заметить что 1 импульс во время периода соответствует принятому «0», а 2 импульса принятой «1». Модуль частотного приемника FrequencyDemodulator.py.

Исходный входной сигнал и соответствующий ему выходной изображены на рис. 6.11 и рис. 6.12

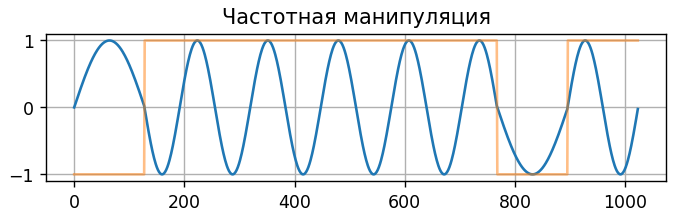


Рисунок 6.11 – Исходный манипулированный сигнал

sb = [[1, 2, 2, 2, 2, 2, 1, 2],

[0, 1, 1, 1, 1, 1, 0, 1]]

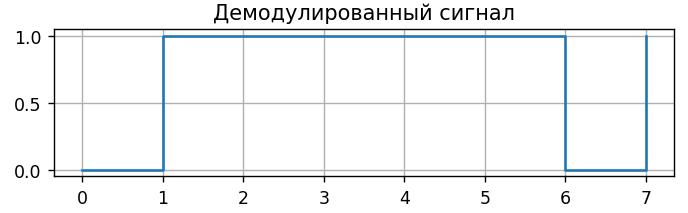


Рисунок 6.12 – Демодулированный сигнал

Проверка на помехоустойчивость будет проходить образом подобному

описанному в пункте 5

ss = [i + random.uniform(0, 0.5) - random.uniform(0, 0.5) for i in s]

sb = [[1, 2, 2, 2, 2, 2, 1, 2],

[0, 1, 1, 1, 1, 1, 0, 1]]

Манипулированный сигнал с помехами изображён на рис. 6.13.

Демодулированный сигнал изображён на рис. 6.14 (совпадает со случаем без помех).

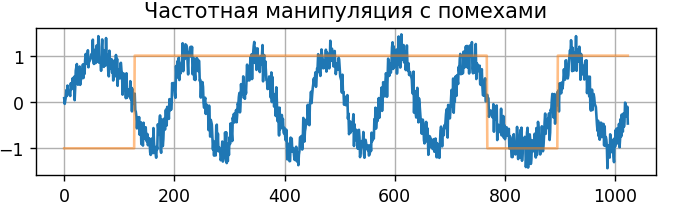


Рисунок 6.13 – Манипулированный сигнал с помехами

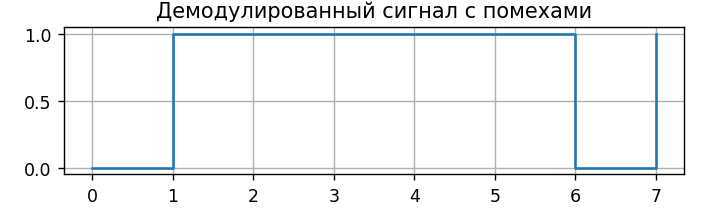


Рисунок 6.14 – Демодулированный сигнал с помехами

# **7 Заключение**

В ходе выполнения лабораторной работы были получены навыки построения АИМ и ШИМ сигнала.

# **8 Ответы на контрольные вопросы**

ТУДУ

# **9 Приложение А**

Файл FrequencyDemodulator.py

def fd(s, K, N, Fd):

b = [[i for i in range(0, int(K / N))],

[i for i in range(0, int(K / N))]]

for i in range(0, int(K / N)):

t = 0

state = 0

for k in range(N \* i, N \* (i + 1)):

newState = getNewState(s[k], Fd)

t = t + 1 if newState != 0 and (state == 0 or newState != state) else t

state = newState if newState != 0 else state

b[0][i] = t

b[1][i] = 1 if t > 1 else 0

return b

def getNewState(sk, Fd):

if sk > Fd:

return 1

elif sk < -Fd:

return -1

else:

return 0

Файл FrequencyModulation.py

import math

import numpy

def frequency(data, A, fsin, k, C, N, Wd):

c = 0

s = []

for i in k:

c = c + math.pi \* C \* Wd \* data[int(numpy.floor(i / N))]

s.append(A \* math.sin(fsin \* math.pi \* i \* C + c))

return s

Файл PhaseDemodulator.py

import numpy

def pd(s, L, K, N, Pd):

b = [[i for i in range(0, int(K / N))],

[i for i in range(0, int(K / N))]]

for i in numpy.arange(0, K / N):

t = 0

for k in numpy.arange(N \* i, N \* (i + 1)):

if abs(s[int(k)] - L[int(k)]) > Pd:

t = t + 1

b[0][int(i)] = t

b[1][int(i)] = 1 if t > 25 else 0

return b

Файл PhaseManipulation.py

import math

import numpy

def phase(data, A, fsin, k, C, N):

s = []

for i in k:

s.append(A \* math.sin(fsin \* math.pi \* i \* C + data[int(numpy.floor(i / N))] \* (math.pi / 2)))

return s

Файл lab2.py

import matplotlib.pyplot as plt

import math

import numpy

import PhaseManipulation

import FrequencyModulation

import PhaseDemodulator

import FrequencyDemodulator

import random

from scipy.fft import fft

# Входные данные

# Наш варик !!!

data = [0, 1, 1, 1, 1, 1, 0, 1]

Pd = 0.39

Fd = 0.37

# Пример

# data = [0, 1, 0, 1, 1, 1, 0, 1]

# Pd = 0.3

# Fd = 0.35

dataLng = len(data)

N = 2 \*\* 7

fsin = 10 \*\* 3

Wd = 10 \*\* 3

Td = math.pi / 2

K = N \* dataLng

k = numpy.arange(0, K)

Tsign = dataLng / K

C = Tsign / fsin

A = 1

t = numpy.arange(0, Tsign - C, C)

noliki = [-1] \* 128

adinichki = [1] \* 128

final = []

for l in data:

if l == 0:

final.extend(noliki)

else:

final.extend(adinichki)

ok = []

for i in k:

ok.append(A \* math.sin(fsin \* math.pi \* i \* C))

s = PhaseManipulation.phase(data, A, fsin, k, C, N)

plt.figure()

plt.subplot(411)

plt.plot(ok, 'r--')

plt.plot(s)

plt.plot(final, alpha=0.5)

plt.title('Фазовая манипуляция')

plt.grid(True)

plt.subplot(412)

s = fft(s)

plt.plot(abs(s)[:100])

plt.title('Спектр фазовой манипуляции')

plt.grid(True)

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

s = FrequencyModulation.frequency(data, A, fsin, k, C, N, Wd)

plt.subplot(413)

plt.plot(ok, 'r--')

plt.plot(s)

plt.plot(final, alpha=0.5)

plt.title('Частотная манипуляция')

plt.grid(True)

plt.subplot(414)

s = fft(s)

plt.plot(abs(s)[:100])

plt.title('Спектр частотной манипуляции')

plt.grid(True)

plt.show()

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

plt.figure('2')

s = PhaseManipulation.phase(data, A, fsin, k, C, N)

L1 = [A \* math.sin(fsin \* math.pi \* i \* C) for i in k]

L2 = [A \* math.cos(fsin \* math.pi \* i \* C) for i in k]

sb1 = PhaseDemodulator.pd(s, L1, K, N, Pd)

sb2 = PhaseDemodulator.pd(s, L2, K, N, Pd)

plt.subplot(511)

plt.step([i for i in range(dataLng)], sb1[1], where='post')

plt.title('Демодулированный сигнал')

plt.grid(True)

plt.subplot(512)

plt.step([i for i in range(dataLng)], sb2[1], where='post')

plt.title('Демодулированный сигнал')

plt.grid(True)

ss = [i + random.uniform(0, 0.5) - random.uniform(0, 0.5) for i in s]

sb1 = PhaseDemodulator.pd(ss, L1, K, N, Pd)

sb2 = PhaseDemodulator.pd(ss, L2, K, N, Pd)

print(sb1)

print(sb2)

s = PhaseManipulation.phase(data, A, fsin, k, C, N)

plt.subplot(513)

plt.plot(ok, 'r--')

plt.plot(s)

plt.plot(ss)

plt.plot(final, alpha=0.5)

plt.title('Фазовая манипуляция с помехами')

plt.grid(True)

plt.subplot(514)

plt.step([i for i in range(dataLng)], sb1[1], where='post')

plt.title('Демодулированный сигнал с помехами')

plt.grid(True)

plt.subplot(515)

plt.step([i for i in range(dataLng)], sb2[1], where='post')

plt.title('Демодулированный сигнал с помехами')

plt.grid(True)

plt.show()

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

s = FrequencyModulation.frequency(data, A, fsin, k, C, N, Wd)

sb = FrequencyDemodulator.fd(s, K, N, Fd)

plt.figure('3')

plt.subplot(411)

plt.plot(s)

plt.plot(final, alpha=0.5)

plt.title('Частотная манипуляция')

plt.grid(True)

plt.subplot(412)

plt.step([i for i in range(dataLng)], sb[1], where='post')

plt.title('Демодулированный сигнал')

plt.grid(True)

ss = [i + random.uniform(0, 0.5) - random.uniform(0, 0.5) for i in s]

plt.figure('3')

plt.subplot(413)

plt.plot(ss)

plt.plot(final, alpha=0.5)

plt.title('Частотная манипуляция с помехами')

plt.grid(True)

sb = FrequencyDemodulator.fd(ss, K, N, Fd)

plt.subplot(414)

plt.step([i for i in range(dataLng)], sb[1], where='post')

plt.title('Демодулированный сигнал с помехами')

plt.grid(True)

plt.show()