МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра вычислительной техники

ОТЧЕТ

по производственной практике

Тема: Кодирование и декодирование информации с использованием М-последовательностей

Студент гр. 8305 _____ Панарин А.Е.

Руководитель _____ Чугунов Л.А.

Санкт-Петербург

2021

ЗАДАНИЕ

НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ

Студент Панарин А.Е.

Группа 8305

Тема практики: Кодирование и декодирование информации с использованием М-последовательностей

Задание на практику:

На примере м-последовательностей 4-го порядка изучить их основные параметры. Программно реализовать функцию генерации последовательностей. Сформировать сигнал на основе м-последовательности, перенести его на рабочую частоту. Сгенерировать аддитивную смесь белого шума и временного сигнала. Декодировать полученный сигнал методом согласованной фильтрации.

Сроки прохождения практики: 30.06.2021 – 13.07.2021

Дата сдачи отчета: 14.07.2021

Дата защиты отчета: 14.07.2021

Студент _____ Панарин А.Е.

Руководитель _____ Чугунов Л.А.

АННОТАЦИЯ

В отчёте приводится описание организации, в которой проходила производственная практика. Представлено описание задания, которое необходимо было выполнить. Также приведены примеры решения данного задания.

SUMMARY

The report contains a description of the organization in which the industrial practice took place. A description of the task that had to be completed is provided. Examples of solving this task are also given.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5
1. ОПИСАНИЕ ОРГАНИЗАЦИИ 6
1.1 Общая информация 6
1.2 Место решаемой задачи в общей проблематике 6
2. ЗАДАНИЕ ДЛЯ ПРАКТИКИ 7
2.1. Задание №1 7
2.2. Задание №2 7
2.3. Задание №3 7
2.4. Задание №4 7
3. РЕШЕНИЕ ЗАДАЧ 8
3.1. Задание №1 8
3.2. Задание №2 9
3.3. Задание №3 10
3.4. Задание №4 12
ЗАКЛЮЧЕНИЕ 16
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17
ПРИЛОЖЕНИЕ А ФУНКЦИЯ ГЕНЕРАЦИИ М-ПОСЛЕДОВАТЕЛЬНОСТИ НА C++ 18
ПРИЛОЖЕНИЕ В ГЕНЕРАЦИЯ СИГНАЛА В MATLAB 20

ВВЕДЕНИЕ

Целью производственной практики является приобретение практических и теоретических навыков, которые полезны при работе на предприятии. Основной задачей было формирование и декодирование полученного сигнала с помощью MATLAB.

1. ОПИСАНИЕ ОРГАНИЗАЦИИ

1.1 Общая информация

АО "Концерн «ЦНИИ „Электроприбор“»" — ведущий институт России в области высокоточной навигации, гироскопии и гравиметрии. Институт имеет статус Государственного научного центра Российской Федерации (ГНЦ РФ).

Основные направления работы института:

- морская навигационная техника;
- электронно-картографическое программное обеспечение и системы;
- инерциальные навигационные системы, гироскопические приборы, системы широкого применения для морской навигации и управления движением судов;
- системы ориентации космических аппаратов;
- антенно-фидерные и коммутационные устройства;
- автоматизированные комплексы радиосвязи;
- приборы точной электромеханики;
- разработка и изготовление средств связи для судов и подводных лодок.

ЦНИИ «Электроприбор» сотрудничает как с отечественными организациями (например, ВМФ России), так и с зарубежными фирмами США, Германии, Индии, Китая, Норвегии, Кореи, Алжира, Финляндии и Японии.

1.2 Место решаемой задачи в общей проблематике

Связь с подводными лодками, обитаемыми и необитаемыми подводными аппаратами, водолазами, когда они находятся в погружённом состоянии — достаточно сложная техническая задача. Основная проблема состоит в том, что электромагнитные волны с частотами, используемыми в традиционной радиосвязи, сильно ослабляются при прохождении через толстый слой проводящего материала, которым является солёная морская вода. Поэтому в ряде

случаев используется звукоподводная связь, осуществляемая в водной среде посредством излучения и приёма модулированных акустических волн.

2. ЗАДАНИЕ ДЛЯ ПРАКТИКИ

2.1. Задание №1

Освоить следующую теорию: принципы формирования псевдослучайных последовательностей (ПСП) максимальной длины – м-последовательностей (Maximum Length Sequence, MLS); порядок м-последовательности; порождающий полином; длина; начальное заполнение регистров; циклический временной сдвиг (ЦВС).

2.2. Задание №2

Программно реализовать функцию формирования м-последовательности с заданными параметрами.

2.3. Задание №3

Кодирование информации значением ЦВС. Формирование таблицы соответствия: номер ЦВС – символ. Формирование сигнала на основе м-последовательности. Перенос на рабочую частоту. Генерация аддитивной смеси белого шума и временного сигнала на основе заданной м-последовательности с фиксированным содержанием.

2.4. Задание №4

Декодирование сигнала на основе м-последовательности. Корреляционная обработка. Метод согласованной фильтрации. Произвести декодирование информации из заданного сигнала.

3. РЕШЕНИЕ ЗАДАЧ

3.1. Задание №1

М-последовательность — псевдослучайная двоичная последовательность, порожденная регистром сдвига с линейной обратной связью и имеющая максимальный период.

Для генерации М-последовательности с периодом $M=2^n-1$ используется примитивный полином $h(x)$ степени n с коэффициентами GF(2), т. е.

$$h(x) = \sum_{i=0}^{n} h_i x^i,$$

где $h_0 = h_n = 1$, а $h_i = \{0,1\}$ при $0 < i < n$.

Примитивные полиномы существуют для всех $n \geq 1$. В табл. 1 приведены полиномы $h(x)$ для n =1…16, которые имеют минимальное число ненулевых коэффициентов h_i и могут быть использованы для генерации соответствующих М-последовательностей

Таблица 1

n	$h(x)$	$M=2^n-1$	n	$h(x)$	$M=2^n-1$
1	$x+1$	1	9	x^9+x^4+1	511
2	x^2+x+1	3	10	$x^{10}+x^3+1$	1023
3	x^3+x+1	7	11	$x^{11}+x^2+1$	2047
4	x^4+x+1	15	12	$x^{12}+x^6+x^4+x+1$	4095
5	x^5+x^2+1	31	13	$x^{13}+x^4+x^3+x+1$	8191
6	x^6+x+1	63	14	$x^{14}+x^{10}+x^6+x+1$	16383
7	x^7+x^3+1	127	15	$x^{15}+x+1$	32787
8	$x^8+x^4+x^3+x^2+1$	255	16	$x^{16}+x^{12}+x^3+x+1$	65535

Известно, что для конкретного значения n существует точно

$$N = \frac{\Phi(M = 2^n - 1)}{n}$$

различных полиномов h(x), являющихся примитивными. Функция Ф(M), называемая функцией Эйлера, представляет собой количество положительных целых чисел, меньших или равных М и взаимно простых с М.

Начальные значения символов a_0, a_1, ..., a_{n-1} М-последовательности могут выбираться произвольно, за исключением нулевой комбинации. Оставшиеся символы вычисляются по рекуррентному выражению

$$a_i = h_1 a_{i-1} \oplus h_2 a_{i-2} \oplus \ldots \oplus h_n a_{i-n}$$

Для каждой М-последовательности с периодом М существует ровно М различных циклических сдвигов.

3.2. Задание №2

Для решения этого задания была написана функция на C++. Данный язык очень популярен и востребован. Примеры работы функции приведены на рисунках 1 и 2.

```
PS C:\Users\alex3\Desktop\Study\practice> .\task2.exe
Enter sequence order (1 <= n <= 16): 4
Enter reg[0]: 1
Enter reg[1]: 1
Enter reg[2]: 1
Enter reg[3]: 1

Enter shift: 0

Result: 111101011001000
```

Рисунок 1. Задание 2

```
PS C:\Users\alex3\Desktop\Study\practice> .\task2.exe
Enter sequence order (1 <= n <= 16): 5
Enter reg[0]: 1
Enter reg[1]: 0
Enter reg[2]: 1
Enter reg[3]: 1
Enter reg[4]: 1

Enter shift: 2

Result: 10101110110001111100110100100000
```

Рисунок 2. Задание 2

3.3. Задание №3

Для формирования сигнала на основе м-последовательности был выбран порождающий полином 4-го порядка: $x^4 + x + 1$. Начальное заполнение регистров: 1111 ($N = 2^4 - 1 = 15$ элементов).

Попробуем передать сообщение со следующим содержанием: TEST_MESSAGAE

Формирование информационного сигнала на основе м-последовательности:

1) Строится элементарный импульс м-последовательности вида:

$$I(t) = \sin(2\pi \cdot f \cdot t)$$

где f – несущая частота сигнала. Берем её равной 5000 Гц. Частота дискретизации сигнала 44100 Гц. Длительность элементарного импульса τ – 10 периодов. Амплитуда импульса равна 1.

2) Сигнал на основе м-последовательности C(t) представляет собой непрерывную последовательность N элементарных импульсов. Длительность всего сигнала равна τ*N. Начальная фаза каждого импульса определяется знаком соответствующего элемента м-последовательности. Если элемент равен 1, то импульс формируется как $I(t) = \sin(2\pi*f*t)$. Если элемент равен 0, то $I(t) = -\sin(2\pi*f*t)$.

3) Пункты 1 и 2 повторяются для каждого символа, передающегося в сообщении. Каждому символу соответствует свой ЦВС исходной последовательности. Соответствие приведено в таблице 2.

Таблица 2

Номер ЦВС	Символ
1	T
2	E
3	S
4	_
5	M
6	A
7	G
8	-
9	-
10	-
11	-
12	-
13	-
14	-
15	-

Соответственно последовательность сдвигов в передаваемом сигнале: 1 2 3 1 4 5 2 3 3 6 7 2

В итоге получается сигнал S(t), несущий в себе всё сообщение.

4) Формируется итоговый сигнал U(t) = S(t) + G(t), где G(t) Гауссов белый шум с нулевым мат. ожиданием и σ=1. Структура сигнала представлена на рисунке 3.

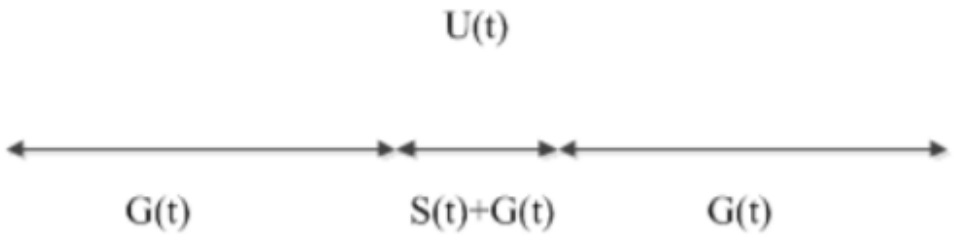

Рисунок 3. Структура сигнала.

В итоге получаем сгенерированный сигнал, рисунок 4.

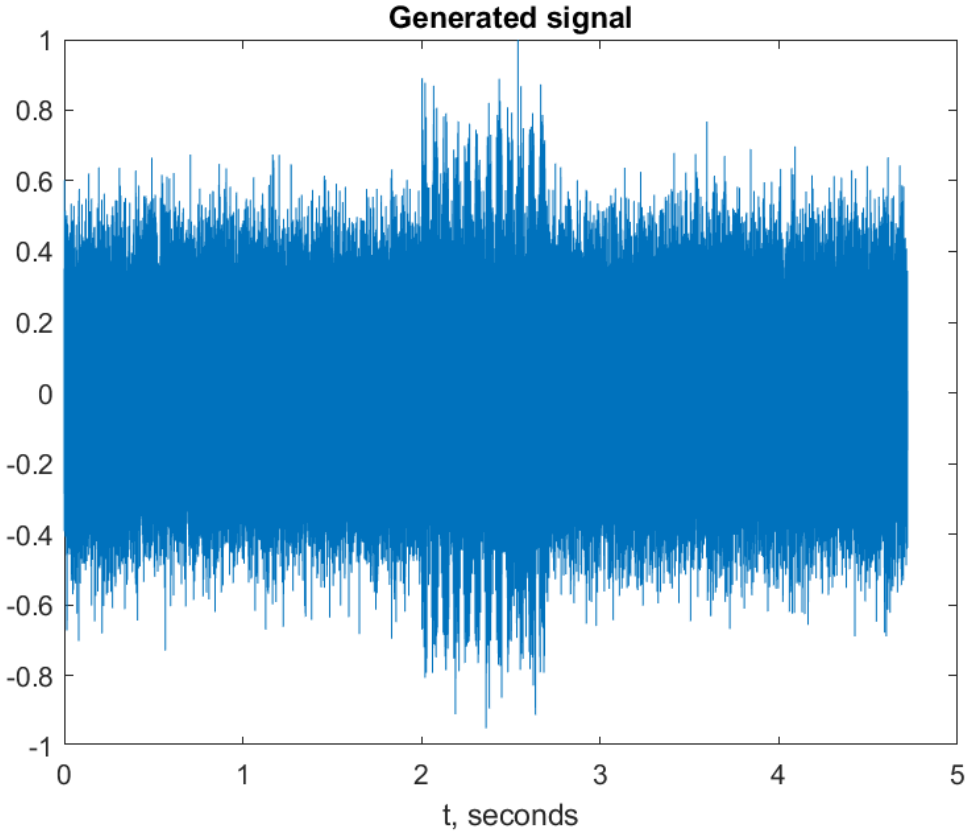

Рисунок 4. Сгенерированный сигнал.

3.4. Задание №4

Исходными данными для формирования импульсной реакции канала (ИРК) являются:

− временной сигнал на выходе приемного тракта U(n), представляющий собой аддитивную смесь информационного сигнала S(n) и внешнего шума среды N(n).

$$U(n) = S(n) + N(n),$$

где n – номер временного отсчета. Данный сигнал сгенерирован в ходе предыдущего задания с частотой дискретизации Fd = 44100 Гц.

− массивы значений опорных сигналов $C_{0_i}(n)$. $C_{0_i}(n)$ представляют собой точную копию информационных сигналов $C_i(n)$. Длительность $C_{0_i}(n)$ равна Ts = N*τ сек., где N – число элементарных импульсов (элементов м-последовательности), τ – длительность элементарного импульса.

Процедура построения ИРК (или согласованной фильтрации) основана на вычислении взаимно-корреляционной функции (ВКФ) входного сигнала U(n) и опорных сигналов $C_{0_i}(n)$. ВКФ вычисляется с использованием процедуры Быстрого Преобразования Фурье (БПФ) и производится в следующем порядке:

1) Определение размерности массива БПФ

   Размерность массива БПФ Nfft выбирается равной степени числа 2, с условием, что опорный сигнал $C_{0_i}(n)$ помещается в половину данного массива. Так, если длительность сигнала составляет Ns = 1300 отсчетов, то Nfft = 4096.

2) Формирование спектра для каждого опорного сигнала

   Формируется исходный массив значений для процедуры БПФ G(n) размерностью Nfft (n = 0,…, Nfft-1) значений, где первые N_s отсчетов равны значениям опорного сигнала $C_{0_i}(n)$, а остальные равны 0 (рисунок 5)

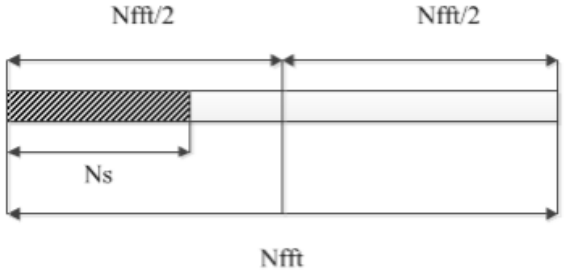

Рисунок 5. Заполнение исходного массива для вычисления спектра опорного сигнала.

   Над массивом G(n) выполняется процедура БПФ с формированием массивов $F_{0_i}(m)$, из Nfft (m = 0,…, Nfft-1) комплексных спектральных значений. Данный массивы $F_{0_i}(m)$ неизменны и используются далее для вычислений.

3) Формирование спектра взаимно-корреляционной функции.

   Входной сигнал U(n) разбивается на окна (выборки) длинной Nfft значений. Окна берутся с половинным перекрытием (рисунок 6).

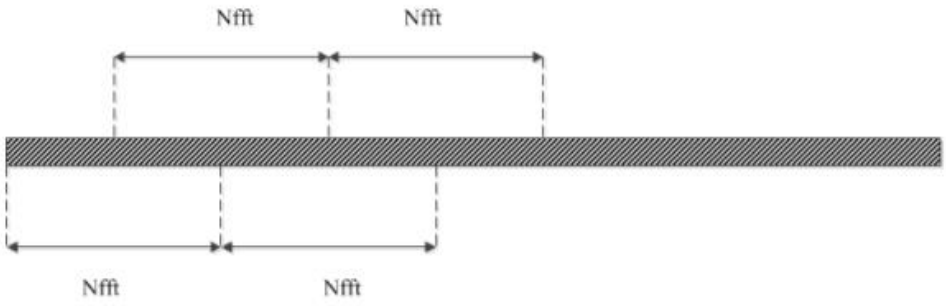

Рисунок 6. Последовательность формирования окон БПФ по данным входного сигнала

Для каждой выборки входного сигнала (окна длительностью Nfft отсчетов) $U_j(n)$ (где j – номер окна) выполняется процедура БПФ с формированием массива $F_j(m)$ из Nfft (m = 0,…, Nfft-1) комплексных спектральных значений (формирование спектра выборки входного сигнала).

Отсчеты спектра ВКФ $F_{вкф}(m)$ формируются как результат умножения значения спектрального отсчета входного сигнала $F_i(m)$ на комплексно-сопряженное значение соответствующего спектрального отсчета опорного сигнала $F_{0_i}(m)$.

$$F_{вкф_i}(m) = F_j(m) * conj(F_{0_i}(m)),$$

где conj(…) – операция комплексного сопряжения, m = 0,…, Nfft-1

4) Формирование ИРК

После формирования спектра ВКФ для текущего окна (расчета Nfft комплексных значений $F_{вкф}(m)$) вторая половина спектра обнуляется (вещественные и мнимые части последних Nfft/2 значений приравниваются 0). Над полученным массивом выполняется процедура обратного БПФ, с формированием массива $R_j(n)$ из Nfft комплексных значений (j – номер окна). Для первых Nfft/2 комплексных значений массива $R_j(n)$ вычисляется квадрат амплитуды $G_j(n)$

$$G_j(n) = \left\| R_j(n) \right\|^2$$

Эти операции выполняются для каждого массива опорных сигналов $C_{0_i}(n)$.

В итоге для каждого окна получаем 15 выходов согласованного фильтра (ИРК). Из них выбираем выход, где значение ИРК достигает максимального значения, и запоминаем номер этого выхода. Этот номер будет соответствовать номеру ЦВС последовательности для текущего окна. Соответственно мы можем декодировать, какой символ передавался в данном окне.

В конце применяется фильтрация, чтобы отбросить окна, где сигнал не передавался вовсе.

Результат декодирования сигнала представлен на рисунке 7.

```
>> name_detection_task

res =

    'TEST_MESSAGAE'
```

Рисунок 7. Результат декодирования

ЗАКЛЮЧЕНИЕ

В ходе прохождения практики я знакомился с задачами, которые могут решаться в концерне «ЦНИИ Электроприбор». Мною были реализованы процессы формирования и декодирования звукового сигнала, сформированного на основе м-последовательностей. Данный способ применяется в звукоподводной связи между кораблём и подводным аппаратом. Были получены теоретические и практические навыки в формировании сигналов на основе м-последовательностей. Поставленное задание было реализовано с помощью C++ и MATLAB.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. И. Д. Захаров, А. А. Ожиганов. Использование порождающих полиномов М-последовательностей при построении псевдослучайных кодовых шкал // Санкт-Петербургский национальный исследовательский университет ИТМО, 2011. С. 1–3.
2. Официальный сайт АО "Концерн «ЦНИИ „Электроприбор“»" // URL: http://www.elektropribor.spb.ru/ (дата обращения 05.07.2021г).
3. Официальный сайт с документацией MATLAB // URL: https://docs.exponenta.ru/matlab/index.html (дата обращения 01.07.2021г)

ПРИЛОЖЕНИЕ А
ФУНКЦИЯ ГЕНЕРАЦИИ М-ПОСЛЕДОВАТЕЛЬНОСТИ НА С++

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>

using namespace std;

vector<int> m_seq_generator(vector<int> poly, vector<int> reg, int n, int k) {
    int result = 0;
    int size = (pow(2, n) - 1);

    //формирование последовательности
    for (int i = n; i < size; i++) {
        for (int j = 1; j <= n; j++) {
            result ^= poly.at(j) * reg.at(i - j);
        }
        reg.push_back(result);
        result = 0;
    }

    //ЦВС
    k = k % size;
    int temp;
    while (k--) {
        temp = reg.at(size - 1);
        for (int i = size - 1; i > 0; --i)
            reg.at(i) = reg.at(i - 1);
        reg.at(0) = temp;
    }

    return reg;
}

int main() {
    int buffer;
    int n; //порядок последовательноти
    int k; //ЦВС
    vector<int> reg; //начальные регистры

    //порождающие полиномы
    vector<vector<int>> polies = {
        {1, 1},
        {1, 1, 1},
        {1, 1, 0, 1},
        {1, 1, 0, 0, 1},
        {1, 0, 1, 0, 0, 1},
        {1, 1, 0, 0, 0, 0, 1},
        {1, 0, 0, 1, 0, 0, 0, 1},
        {1, 0, 1, 1, 1, 0, 0, 0, 1},
        {1, 0, 0, 0, 1, 0, 0, 0, 0, 1},
        {1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1},
        {1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1},
        {1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1},
        {1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1},
        {1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1},
        {1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
```

```
        {1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1}
    };

    cout << "Enter sequence order (1 <= n <= 16): ";
    cin >> n;
    reg.reserve(pow(2, n) - 1);

    for(int i = 0; i < n; i++) {
        cout << "Enter reg[" << i << "]: ";
        cin >> buffer;
        reg.push_back(buffer);
    }
    cout << endl;

    cout << "Enter shift: ";
    cin >> k;
    k = (k < 0) ? k + (pow(2, n) - 1) : k;

    reg = m_seq_generator( polies[n - 1], reg, n, k);

    cout << endl << "Result: ";
    for_each(reg.begin(), reg.end(), [](int i) {
            cout << std::left << i;
        });

    return 0;
}
```

ПРИЛОЖЕНИЕ В
ГЕНЕРАЦИЯ СИГНАЛА В MATLAB

m_generator.m

```
%m-sequence generation
function m_seq = m_generator(polynom, registers, n, k)
    size = power(2, n) - 1;

    for i = n:(size - 1)
        registers(end + 1) = 0;
        for j = 1:n
            registers(end) = xor(registers(end), polynom(j + 1) * registers(i -
j + 1));
        end
    end

    k = mod(k, size);
    new_reg = circshift(registers, k);

    m_seq = new_reg;
end
```

I.m

```
%impulse generation
function impulse = I(tStart, tEnd, sign, frequencyOfSignal, frequencyOfDiscret,
a)
    time = tStart: 1 / frequencyOfDiscret : (tEnd - (1 / frequencyOfDiscret));
    impulse = a * (-1) * power(-1, sign) * sin(2 * pi * frequencyOfSignal *
time);
end
```

signal_generator.m

```
function value = signal_generator(mSeq, N, tau, frequencyOfSignal, frequencyOfD,
amplitude)
    value = [];

    for j=1:15
        temp_arr = [];
        temp = mSeq(j,:);
        for i = 1 : N
            temp_arr = [temp_arr, I(2 + (j-1)*0.03 + 1 / frequencyOfSignal * tau
* (i - 1), 2 + (j-1)*0.03 + 1 / frequencyOfSignal * tau * i, temp(i),
frequencyOfSignal, frequencyOfD, amplitude)];
        end
        value = [value; temp_arr];
    end
end
```

message_detection.m

```
keySet = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15];
valueSet = {'T', 'E', 'S', '_', 'M', 'A', 'G', '-', '-', '-', '-', '-', '-', '-
', '-'};
M = containers.Map(keySet,valueSet);
%1+x^1+x^4
polynom = [1, 1, 0, 0, 1];
registers = [1, 1, 1, 1];
frequencyOfSignal = 5000;
frequencyOfD = 44100;
tau = 10;
SNR = 6;
amplitude = 1;
powerOfPoly = 4;
N = power(2, powerOfPoly) - 1;
Nfft = 4096;

mSeq = [];
for i=0:14
   mSeq = [mSeq; m_generator(polynom, registers, powerOfPoly, i)];
   mSeq(i+1,:);
end

%noise generation
countOfMeasurement = 2 * frequencyOfD + 1;
valueBeforeSignal = zeros(1, countOfMeasurement);
valueBeforeSignal = awgn(valueBeforeSignal, SNR);
valueAfterSignal = zeros(1, countOfMeasurement);
valueAfterSignal = awgn(valueAfterSignal, SNR);
noiseCount = round(1 / frequencyOfSignal * tau * frequencyOfD * N + 1);
valueBetweenSignal = zeros(1, noiseCount);

% forming a signal
values = signal_generator(mSeq, N, tau, frequencyOfSignal, frequencyOfD,
amplitude);
value = [];
value = [value, values(1,:)];
value = [value, valueBetweenSignal];
value = [value, values(2,:)];
value = [value, valueBetweenSignal];
value = [value, values(3,:)];
value = [value, valueBetweenSignal];
value = [value, values(1,:)];
value = [value, valueBetweenSignal];
value = [value, values(4,:)];
value = [value, valueBetweenSignal];
value = [value, values(5,:)];
value = [value, valueBetweenSignal];
value = [value, values(2,:)];
value = [value, valueBetweenSignal];
value = [value, values(3,:)];
value = [value, valueBetweenSignal];
value = [value, values(3,:)];
value = [value, valueBetweenSignal];
value = [value, values(6,:)];
value = [value, valueBetweenSignal];
value = [value, values(7,:)];
value = [value, valueBetweenSignal];
value = [value, values(2,:)];
```

```
value = [value, valueBetweenSignal];

%noise overlay
valueDuringSignal = awgn(value, SNR);
resultSignal = [valueBeforeSignal, valueDuringSignal, valueAfterSignal];
%signal normalization
maxValueOfSignal = max(abs(resultSignal));
resultSignal = resultSignal / maxValueOfSignal;

%output
t = 0 : 1/frequencyOfD : 4 + 1 / frequencyOfSignal * tau * N * 12 * 2;
t = t(1 : length(resultSignal));
figure
plot(t, resultSignal)
title('Generated signal')
xlabel('t, seconds')
saveas(gcf, 'Generated signal', 'png')
audiowrite('output.wav', resultSignal, frequencyOfD);

%----------------signal detection-----------------------------------------
F0 = [];
%filters
for i = 1 : 15
   tmp = [values(i,:), zeros(1, Nfft - length(values(i,:)))];
   tmp = fft(tmp);
   F0 = [F0; tmp];
end
resultSignalForDetect = [resultSignal, zeros(1, Nfft - mod(length(resultSignal),
Nfft))];
iter = length(resultSignalForDetect)/Nfft;

GMaxGlobal = [];
IndGlobal = [];
for i = 0 : (iter - 1) * 2
    GMax = 0;
    for j = 1 : 15
        G = [];
        U = resultSignalForDetect((i/2 * Nfft + 1) : ((i/2 + 1) * Nfft));
        F = fft(U);
        Fvkf = F .* conj(F0(j,:));
        Fvkf(Nfft/2 + 1 : Nfft) = 0 ;
        R = ifft(Fvkf, 'symmetric');
        R = R(1 : Nfft/2);
        G = [G, R.^2];
        if max(G) > GMax
            GMax = max(G);
            IndGMax = j;
        end
    end
    GMaxGlobal = [GMaxGlobal, GMax];
    IndGlobal = [IndGlobal, IndGMax];
end

%output
res = [];
for i = 1:length(GMaxGlobal)
    if GMaxGlobal(i) > max(GMaxGlobal) * 0.80
        res = [res, M(IndGlobal(i))];
    end
```

```
end

res
```