

Федеральное государственное бюджетное образовательное учреждение  
высшего образования «Сибирский государственный университет  
телекоммуникаций и информатики»

## **Лабораторная работа №4**

**по дисциплине «Основы систем мобильной связи»**

**«Изучение корреляционных свойств последовательностей,  
используемых для синхронизации в сетях мобильной связи»**

Выполнил:

Шаповал Н.О.

Группа: ИА-232

Проверил: Дроздова В.Г.

Вариант: 16

GitHub:

<https://github.com/nikin1/osms>



Новосибирск 2024

## Содержание

Цель работы.....	3
Краткие теоретические сведения.....	4
Этапы выполнения работы.....	10
Контрольные вопросы.....	18
Заключение .....	20

## **Цель работы**

Получить представление о том, какие существуют псевдослучайные двоичные последовательности, какими корреляционными свойствами они обладают и как используются для синхронизации приемников и передатчиков в сетях мобильной связи.

## **Задачи работы:**

- Написать программу на языке C/C++ для генерации последовательности Голда;
- Вывести получившуюся последовательность на экран;
- Сделать поэлементный циклический сдвиг последовательности и посчитать автокорреляцию исходной последовательности и сдвинутой;
- Сформировать таблицу с битовыми значениями последовательностей, в последнем столбце которой будет вычисленное значение автокорреляции;
- Сформировать еще одну последовательность Голда, используя свою схему, такую, что  $x=x+1$ , а  $y=y-5$ ;
- Вычислить значение взаимной корреляции исходной и новой последовательностей;
- Прodelать предыдущие шаги в Matlab, используя функции `xcorr()` и `autocorr()` для вычисления соответствующих корреляций;
- Выведите на график в Matlab функцию автокорреляции в зависимости от величины задержки (lag).

## Краткие теоретические сведения

### *Псевдослучайные двоичные последовательности*

Псевдослучайные двоичные последовательности (PN-sequences – PseudoNoise) – это частный случай псевдослучайных последовательностей, элементами которой являются только 2 возможных значения (1 и 0 или -1 и +1). Такие последовательности очень часто используются в сетях мобильной связи. Возможные области применения:

- оценка вероятности битовой ошибки (BER – Bit Error Rate). В этом случае передатчик передает приемнику заранее известную PN-последовательность бит, а приемник анализируя значения бит на конкретных позициях, вычисляет количество искаженных бит и вероятность битовой ошибки в текущих радиоусловиях, что затем может быть использовано для работы алгоритмов, обеспечивающих помехозащищенность системы;
- временная синхронизация между приемником и передатчиком.

Включаясь, абонентский терминал начинает записывать сигнал, дискретизируя его с требуемой частотой, в результате чего формируется массив временных отсчетов и требуется понять, начиная с какого элемента в этом массиве собственно содержатся какие-либо данные, как именно структурирована ось времени, где начинаются временные слоты. Используя заранее известную синхронизирующую PN-последовательность (синхросигнал), приемник сравнивает полученный сигнал с этой последовательностью на предмет «сходства» - корреляции. И если фиксируется корреляционный пик, то на стороне приема можно корректно разметить буфер с отсчетами на символы, слоты, кадры и пр.

- расширение спектра. Используется для повышения эффективности передачи информации с помощью модулированных сигналов через канал с сильными линейными искажениями (замираниями), делая систему устойчивой к узкополосным помехам (например, в 3G WCDMA).

Псевдослучайная битовая последовательность должна обладать следующими свойствами, чтобы казаться почти случайной:

1. Сбалансированность (balance), то есть число единиц и число нулей на любом интервале последовательности должно отличаться не более чем на одну.
2. Цикличность. Циклом в данном случае является последовательность бит с одинаковыми значениями. В каждом фрагменте псевдослучайной битовой последовательности примерно половину составляли циклы длиной 1, одну четверть – длиной 2, одну восьмую – длиной 3 и т.д.
3. Корреляция. Корреляция оригинальной битовой последовательности с ее сдвинутой копией должна быть минимальной. Автокорреляция этих последовательностей – это практически дельта-функция во временной области, как для аддитивного белого гауссовский шума AWGN (Additive white Gaussian noise), а в частотной области – это константа.

Как можно сгенерировать последовательность, обладающую вышеперечисленными свойствами?

Для этого можно использовать, например, линейный четырехразрядный регистр сдвига с обратной связью, сумматора по модулю 2 и контуром обратной связи со входом регистра [3]. Работа регистра тактируется синхрои́мпульсами и с каждым новым тактом осуществляется сдвиг битовой последовательности вправо, а содержимое регистров 3 и 4 суммируется по модулю два, при этом результат суммирования подается на вход регистра 1, как показано на рисунке 4.1



Рис. 4.1. Пример способа формирования псевдослучайной битовой последовательности.

Рассмотрим пример формирования псевдослучайной битовой последовательности с помощью схемы, показанной на рисунке 4.1, при условии, что регистр проинициализирован последовательностью 1 0 0 0. На каждом такте эта последовательность будет сдвигаться на одну позицию вправо, при этом на выходе будут появляться биты псевдослучайной последовательности. В таблице 4.1 показаны состояния разрядов регистра на каждом такте и выходные биты.

Табл. 4.1. Формирование псевдослучайной битовой последовательности.

1	2	3	4	Выход
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
1	0	0	1	1
1	1	0	0	0
0	1	1	0	0

1	0	1	1	1
0	1	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
1	0	0	1	1

На выход всегда идут биты из 4-го разряда регистра. Очевидно, что длина полученной последовательности равна  $2^m - 1 = 15$  – максимальное число различных состояний нашего регистра, где  $m=4$  – число разрядов в сдвиговом регистре, используемом для формирования последовательности, а затем, начиная с 16-го бита, значения на выходе начинают циклически повторяться. Такие последовательности еще называются  $m$ -последовательностями (от англ. слова maximum – последовательности максимальной длины). Важно заметить, что инициализирующая битовая последовательность (или полином) не может быть нулевой, так как из всех нулей невозможно создать последовательность, содержащую единицы, данным способом.

Проанализируем последовательность, полученную в таблице 4.1 с точки зрения наличия свойств псевдослучайных битовых последовательностей:

1. Сбалансированность: 8 единиц и 7 нулей.
2. Цикличность: нет циклов длиннее 4х (1 цикл из 4-х единиц, 1 цикл из 3-х нулей, 2 цикла из нулей и единиц, и 4 цикла длиной, равной одному).

3. Корреляция: автокорреляционная функция периодического сигнала  $x(t)$  с периодом  $T_0$  в нормированной форме (4.1) - (4.2)

$$R_x(\tau) = \frac{1}{K} \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} x(t)x(t+\tau)dt, \quad (4.1)$$

$$\text{где } K = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} x^2(t)dt \quad (4.2)$$

Нормированная автокорреляционная функция псевдослучайного сигнала с длительностью символа, равной единице и периодом  $p=2^m - 1$  может быть определена как (4.3):

$$R_x(\tau) = \frac{1}{p} \cdot \left\{ \begin{array}{l} \text{число различающихся бит} \\ \text{при сравнении оригинальной} \\ \text{и сдвинутой последовательностей} \end{array} \right\} \quad (4.3)$$

Для примера, определим значение автокорреляции последовательности из таблицы 4.1 со сдвигом на 1 элемент.

0 0 0 1 0 0 1 1 0 1 0 1 1 1 1

1 0 0 0 1 0 0 1 1 0 1 0 1 1 1

-----

о о о о о о о о о о о о о о о

о – отличаются; с – совпадают.

Число совпадений: 7;

Число несовпадений: 8.

Следовательно,

$$R_x(\tau = 1) = \frac{1}{15} (7 - 8) = -\frac{1}{15}.$$

Автокорреляция для любого сдвига будет равна  $-1/15$ , и лишь в момент полного совпадения всех элементов будет наблюдаться пик корреляционной функции  $R_x(\tau = 0) = +1$ . На рисунке 4.2 показана автокорреляционная функция псевдослучайной бинарной последовательности.



Рис. 4.2. Автокорреляционная функция псевдослучайной бинарной последовательности в зависимости от величины задержки

Чем длиннее последовательность, тем выше пик ее автокорреляционной функции, и тем больше напоминает дельта-функцию. Такого типа автокорреляцией характеризуется и белый гауссовский шум, поэтому в англоязычной литературе такие последовательности называют *pseudo noise sequences*.

Чем острее автокорреляционный пик (то есть чем длиннее последовательность), тем удобнее использовать данные последовательности для решения проблем синхронизации в сетях мобильной связи.

Действительно, абонентский терминал при начальном включении должен засинхронизировать начало своих временных слотов на временной оси приемника и передатчика. Поэтому обычно базовые станции периодически отправляют специальные синхронизирующие последовательности, в качестве которых часто используются именно *m*-последовательности, и терминал вычисляет автокорреляцию этой заранее известной последовательности с полученным записанным сигналом, и в тот момент, когда фиксируется автокорреляционный пик, абонент отмечает начало слота на своей оси времени (а точнее номер отсчета в буфере, начиная с которого идет передаваемый базовой станцией слот с данными).

Стоит отметить, что даже в случае наличия ошибок в принятой синхропоследовательности, возникших вследствие помех, присутствующих в канале связи, приемник все равно достаточно легко обнаружит явный корреляционный пик.

На рисунке 4.3 представлены варианты реализации схемы синхронизации с помощью последовательного и параллельного поиска

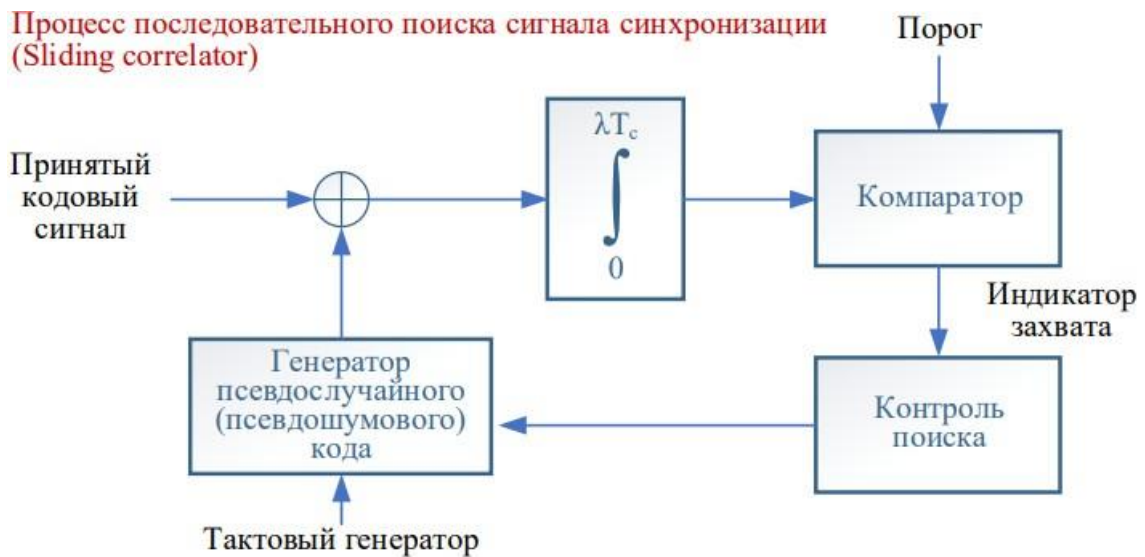
Разновидности псевдо-шумовых битовых последовательностей

*M*-последовательности – не единственные *PN*-последовательности, используемые в системах мобильной связи. Существуют также коды Баркера, коды Голда, коды Касами, коды Уолша-Адамара.

Коды Голда формируются путем суммирования по модулю 2 двух *M*-последовательностей одинаковой длины. Коды Касами также формируются из *M*-последовательностей путем взятия периодических выборок из этих последовательностей и суммированием их по модулю два. Данные коды обладают очень хорошими взаимокорреляционными свойствами.



Процесс последовательного поиска сигнала синхронизации  
(Sliding correlator)



Процесс параллельного поиска сигнала синхронизации

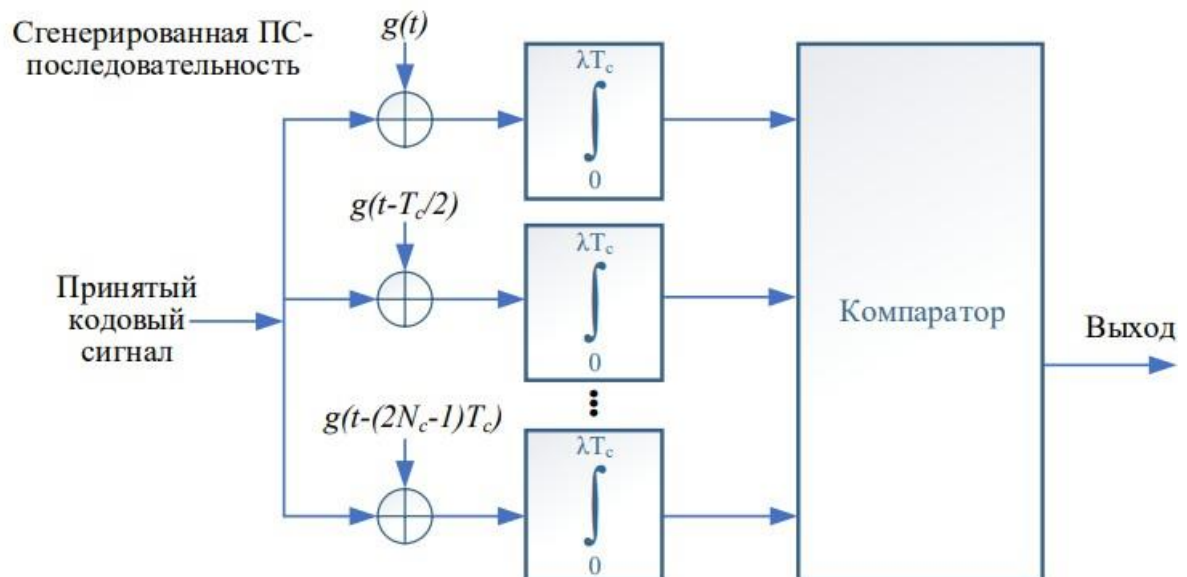
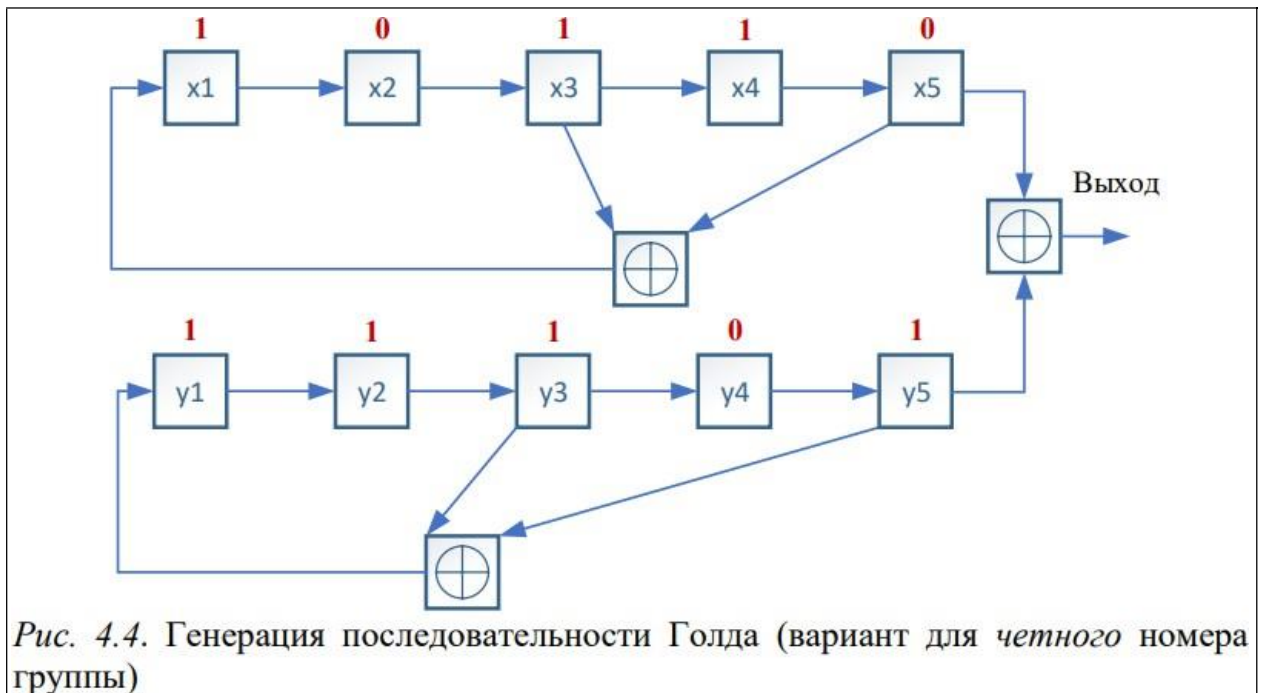


Рис. 4.3. Синхронизация с помощью последовательного и параллельного поиска

## Этапы выполнения работы

### Исходные данные:



$x = 16 = 10000$

$y = 23 = 10111$

1. Сначала требуется написать программу на языке C/C++ для генерации последовательности Голда, используя схему, изображенную на рисунке 4.4. Для выполнения задания будем использовать язык C++. Создадим функцию `create_gold_sequence`, которая будет принимать в себя два вектора:

```
vector<int> create_gold_sequence(vector<int>& x, vector<int>& y){
    vector<int> gold_sequence;
    int xor_shift_x, xor_shift_y;
    int last_bit_x, last_bit_y;
    for (int i = 0; i < LENGTH_SEQUENCE; i++){
        xor_shift_x = x[2] ^ x[4];
        xor_shift_y = y[2] ^ y[4];

        gold_sequence.push_back(x.back() ^ y.back());

        x.pop_back();
        y.pop_back();
        x.insert(x.begin(), xor_shift_x);
```

```

        y.insert(y.begin(), xor_shift_y);

    }

    return gold_sequence;
}

```

Функция составляет последовательность Голда согласно схеме изображённой на рисунке 4.4.

2. Создадим два вектора, которые соответствуют значениям  $x$  и  $y$  и вызовем функцию создания последовательности Голда.

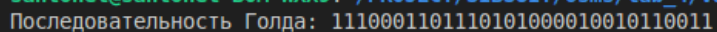
```

vector<int> x = {0, 1, 1, 0, 0};
vector<int> y = {1, 0, 0, 1, 1};

cout << "Последовательность Голда: ";
vector<int> gold_sequence = create_gold_sequence(x, y);
for (int i : gold_sequence){
    cout << i;
}
cout << endl;

```

Получаем следующий результат:



```

Последовательность Голда: 1110001101110101000010010110011

```

3. Далее необходимо сделать поэлементный сдвиг последовательности и посчитать автокорреляцию исходной последовательности и сдвинутой. Реализуем соответствующие функции.

Функция циклического сдвига последовательности:

```

vector<int> shift_sequence(vector<int>& original_sequence, int shift){
    int N = original_sequence.size();
    vector<int> shifted_sequence(N);

    for (int i = 0; i < N; i++){
        shifted_sequence[i] = original_sequence[(i - shift + N) % N];
    }

    return shifted_sequence;
}

```

Функция автокорреляции последовательности:

```
double autoCorrelation(vector<int>& x, vector<int>& y){
    int equal = 0, different = 0;
    double auto_corr;
    for (int i = 0; i < LENGTH_SEQUENCE; i++){
        if (x[i] == y[i]){
            equal += 1;
        }
        else{
            different += 1;
        }
    }

    auto_corr = (1.0 / LENGTH_SEQUENCE) * (equal - different);
    return auto_corr;
}
```

Сформируем таблицу с битовыми значениями последовательности, в последнем столбце которой будет находиться результат вычисления автокорреляции. Для этого также напишем функцию, на вход которой будет поступать последовательность:

```
void print_table(vector<int>& sequence){
    int N = sequence.size();
    cout << "Сдвиг\tПоследовательность\t\tАвтокорреляция" << endl;
    for (int i = 0; i < N + 1; i++){
        vector<int> shifted_sequence = shift_sequence(sequence, i);
        double corr = autoCorrelation(sequence, shifted_sequence);
        cout << i << "\t";
        for (int i : shifted_sequence){
            cout << i << " ";
        }
        cout << "\t";

        cout << corr << endl;
    }
}
```

После вызова функции получаем следующий результат:

Сдвиг	Последовательность	Автокорреляция
0	1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1	1
1	1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1	-0.0322581
2	1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0	-0.0322581
3	0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0	-0.0322581
4	0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1	-0.0322581
5	1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1	-0.0322581
6	1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0	-0.0322581
7	0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1	-0.0322581
8	1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0	-0.0322581
9	0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0	-0.0322581
10	0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1	-0.0322581
11	1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0	-0.0322581
12	0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0	-0.0322581
13	0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0	-0.0322581
14	0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0	-0.0322581
15	0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1	-0.0322581
16	1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0	-0.0322581
17	0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1	-0.0322581
18	1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0	-0.0322581
19	0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1	-0.0322581
20	1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1	-0.0322581
21	1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1	-0.0322581
22	1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0	-0.0322581
23	0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1	-0.0322581
24	1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1	-0.0322581
25	1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0	-0.0322581
26	0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0	-0.0322581
27	0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0	-0.0322581
28	0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1	-0.0322581
29	1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1	-0.0322581
30	1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1	-0.0322581
31	1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1	1

4. Сформируем ещё одну последовательность Голда, где  $x = 17$  (10001) и  $y = 18$  (10010).

Новая последовательность Голда: 1100011011101010000100101100111

5. Вычислим взаимную корреляцию исходной и новой последовательности. Для этого воспользуемся функцией для расчёта нормализованной корреляции из лабораторной работы 3:

```
double NormalizedCorrelation(vector<int>& x, vector<int>& y){
    int N = x.size();
    double corr = 0;
    double sumXY = 0;
    double sumX2 = 0;
```

```

double sumY2 = 0;

for (int i = 0; i < N; i++){
    sumXY += x[i] * y[i];
    sumX2 += x[i] * x[i];
    sumY2 += y[i] * y[i];
}

corr = sumXY / sqrt(sumX2 * sumY2);
return corr;
}

```

Получим результат:

Взаимная корреляция исходной и новой последовательности: 0.5

6. Следующим шагом является повторение всех действий, но только в среде Matlab.

Функция для формирования последовательности Голда:

```

function gold_sequence = create_gold_sequence(x, y, length_sequence)
    gold_sequence = zeros(1, length_sequence);

    for i = 1:length_sequence
        xor_shift_x = bitxor(x(3), x(5));
        xor_shift_y = bitxor(y(3), y(5));

        gold_sequence(i) = bitxor(x(end), y(end));

        x = [xor_shift_x, x(1:end-1)];
        y = [xor_shift_y, y(1:end-1)];
    end
end

```

Сформированная последовательность Голда для исходных данных:

1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 |

Далее сформируем таблицу с битовыми значениями последовательности и результатом вычисления автокорреляции. Для вычисления автокорреляции используется функция autocorr, которая сразу вернёт значения задержки и результат автокорреляции. Для отображения циклического сдвига последовательности используется функция circshift().

```

function print_table(sequence)
N = length(sequence);

fprintf('Сдвиг | Последовательность | Автокорреляция\n');
fprintf('-----|-----|-----\n');

[corr, lags] = autocorr(sequence, 'NumLags', N - 1);

for i = 1:N
shifted_sequence = circshift(sequence, i - 1);

fprintf('%5d | ', lags(i)); fprintf('%d ',
shifted_sequence); fprintf('| %10.4f\n', corr(i));
end

fprintf('\n'); figure;
plot(0:N - 1, corr, '-o'); xlabel('Задержка (lag)');
ylabel('Автокорреляция');
title('Функция автокорреляции в зависимости от задержки');
xlim([0 N])
grid on;

end

```

В итоге выводится таблица:

Сдвиг	Последовательность	Автокорреляция
0	1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1	1.0000
1	1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1	-0.0636
2	1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0	-0.0938
3	0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0	-0.0616
4	0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1	0.0332
5	1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1	0.0655
6	1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0	-0.0315
7	0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1	-0.1284
8	1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0	-0.1586
9	0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0	0.0653
10	0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1	0.0976
11	1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0	-0.0618
12	0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0	-0.1587
13	0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0	0.0652
14	0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0	0.0974
15	0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1	-0.0661
16	1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 0	0.0328
17	0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1	-0.1308
18	1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0	-0.0985
19	0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1 1	0.1254
20	1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 1	0.0285
21	1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1	-0.1309
22	1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0	-0.0987
23	0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1	0.1253
24	1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1	0.0950
25	1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0	-0.0019
26	0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0	-0.0988
27	0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0	-0.0665
28	0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1	0.0282
29	1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1	0.0605
30	1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1	0.0302



Как можно заметить, результат вычисления автокорреляции в Matlab не сходится с результатом на C++. Это связано с тем, что мы использовали функцию `autocorr()`, которая вычисляет автокорреляцию по другой формуле, нежели мы используем при расчёте на C++.

Сформируем новую последовательность Голда и вычислим взаимную корреляцию новой и исходной последовательности при помощи функции `xcorr()`:

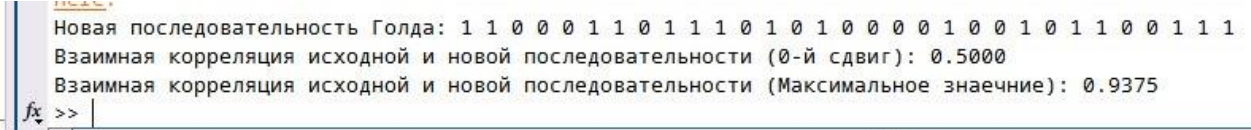
```
x = [0, 1, 1, 0, 1];
y = [0, 1, 1, 1, 0];

new_gold_sequence = create_gold_sequence(x, y, length_sequence);

fprintf('Новая последовательность Голда: ');
fprintf('%d ', new_gold_sequence);
fprintf('\n');

[corr, lags] = xcorr(gold_sequence, new_gold_sequence, 'normalized');
fprintf('Взаимная корреляция исходной и новой последовательности (0-й сдвиг): %.4f\n', corr(length_sequence));
fprintf('Взаимная корреляция исходной и новой последовательности (Максимальное значение): %.4f\n', max(corr));
```

Так как `xcorr()` возвращает взаимные корреляции, сдвигая последовательности, выведем результат на 0-м сдвиге (что считалось на C++) и максимально возможную корреляцию:



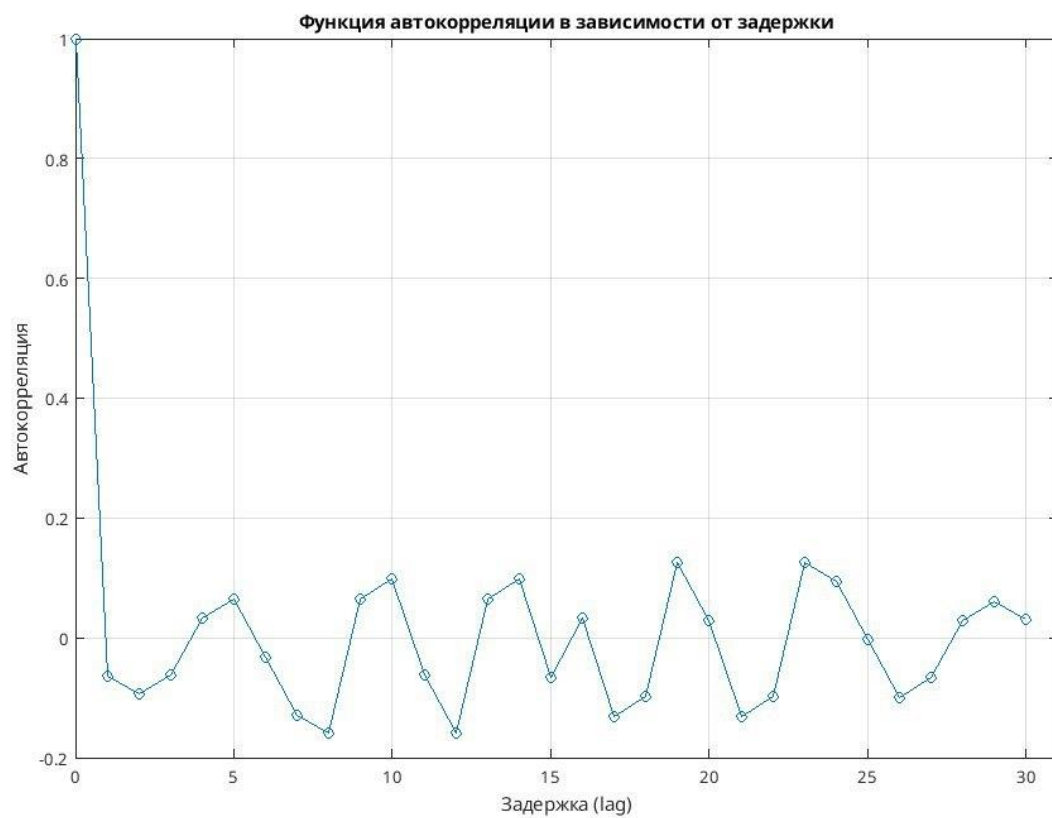
```
Новая последовательность Голда: 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1
Взаимная корреляция исходной и новой последовательности (0-й сдвиг): 0.5000
Взаимная корреляция исходной и новой последовательности (Максимальное значение): 0.9375
```

Как видно, результат на 0-м сдвиге абсолютно идентичен результату, полученному на C++.

7. Выведем на график в Matlab функцию автокорреляции в зависимости от величины задержки (lag):

```
figure;
plot(0:N - 1, corr, '-o');
xlabel('Задержка (lag)');
ylabel('Автокорреляция');
title('Функция автокорреляции в зависимости от задержки');
xlim([0 N])
grid on;
```





## Контрольные вопросы

### 1) Для чего в мобильных сетях могут использоваться псевдослучайные последовательности?

Возможные области применения:

- оценка вероятности битовой ошибки (BER – Bit Error Rate). В этом случае передатчик передает приемнику заранее известную PN-последовательность бит, а приемник анализируя значения бит на конкретных позициях, вычисляет количество искаженных бит и вероятность битовой ошибки в текущих радиоусловиях, что затем может быть использовано для работы алгоритмов, обеспечивающих помехозащищенность системы;
- временная синхронизация между приемником и передатчиком.

Включаясь, абонентский терминал начинает записывать сигнал, дискретизируя его с требуемой частотой, в результате чего формируется массив временных отсчетов и требуется понять, начиная с какого элемента в этом массиве собственно содержатся какие-либо данные, как именно структурирована ось времени, где начинаются временные слоты. Используя заранее известную синхронизирующую PN-последовательность (синхросигнал), приемник сравнивает полученный сигнал с этой последовательностью на предмет «сходства» - корреляции. И если фиксируется корреляционный пик, то на стороне приема можно корректно разметить буфер с отсчетами на символы, слоты, кадры и пр.

- расширение спектра. Используется для повышения эффективности передачи информации с помощью модулированных сигналов через канал с сильными линейными искажениями (замираниями), делая систему устойчивой к узкополосным помехам (например, в 3G WCDMA).

### 2) Что значит положительная корреляция сигналов?

Положительная корреляция сигналов означает, что при увеличении амплитуды одного сигнала увеличивается амплитуда второго сигнала, и наоборот. Это свидетельствует о схожести их временных характеристик и позволяет предполагать, что сигналы имеют общую или взаимозависимую природу.

### **3) Что такое корреляционный прием сигналов?**

### **3) Что такое корреляционный прием сигналов?**

Корреляционный прием — это метод, который использует свойства корреляции для определения наличия или отсутствия полезного сигнала в принятом шумовом сигнале. Применяя к принимаемому сигналу корреляционную функцию, можно выделить моменты, когда входной сигнал совпадает с эталоном, что позволяет извлекать полезный сигнал из шума.

### **4) Как вычисление корреляционных функций помогает синхронизироваться приемнику и передатчику в сетях мобильной связи?**

Вычисление корреляционных функций помогает приемнику распознавать начало и конец передаваемого сигнала, сравнивая его с известным шаблоном. В сетях мобильной связи это необходимо для синхронизации тактовой частоты передатчика и приемника. Используя корреляционные функции, приемник может точно определить временное положение сигнала, что улучшает точность временной синхронизации между узлами сети и обеспечивает стабильность связи.

### **5) Какими свойствами обладают псевдослучайные битовые последовательности?**

1. Сбалансированность (balance), то есть число единиц и число нулей на любом интервале последовательности должно отличаться не более чем на одну.
2. Цикличность. Циклом в данном случае является последовательность бит с одинаковыми значениями. В каждом фрагменте псевдослучайной битовой последовательности примерно половину составляли циклы длиной 1, одну четверть — длиной 2, одну восьмую — длиной 3 и т.д.
3. Корреляция. Корреляция оригинальной битовой последовательности с ее сдвинутой копией должна быть минимальной. Автокорреляция этих последовательностей — это практически дельта-функция во временной области, как для аддитивного белого гауссовский шума AWGN (Additive white Gaussian noise), а в частотной области — это константа.

### **6) Какие разновидности PN-последовательностей вам известны?**

M-последовательности, коды Баркера, коды Голда, коды Касами, коды Уолша-Адамара.

## Заключение

В этой лабораторной работе мы получили представление о том, какие существуют псевдослучайные двоичные последовательности, какими корреляционными свойствами они обладают и как используются для синхронизации приемников и передатчиков в сетях мобильной связи.

Мы разработали функции на языке программирования C++ для создания последовательности Голда и, используя циклический сдвиг, провели автокорреляцию этой последовательности.

Также мы воспользовались MATLAB для сравнения результатов работы встроенной функции автокорреляции с функцией, которая рассчитывает автокорреляцию по специальной формуле. Кроме того, мы построили график зависимости автокорреляции от величины задержки.