МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

Кафедра телекоммуникационных систем и вычислительных средств

(ТС и ВС)

Лабораторная работа № 4

По дисциплине

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

Студент:

Группа № ИА331 Р. К. Рубцов

Преподаватель: В. Г. Дроздова

Новосибирск 2025 г.

Цель занятия:

- 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.

Краткие теоретические сведения:

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

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 2 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.

Автокорреляция: $R_x(\tau = 1) = \frac{1}{P}(c - o)$,

- tau – Лаг,
- p - 2^кол-во разрядом – 1,
- c – совпадения,
- o – отличия.

Порядок выполнения работы:

1. Напишите программу на языке C/C++ для генерации последовательности Голда, используя схему, изображенную на рисунке 4.4, если ваша группа с четным номером и 4.5 – если с нечетным, и порождающие полиномы x и y, при этом x – это ваш порядковый номер в журнале в двоичном формате (5 бит), а y – это x+7 (5 бит).

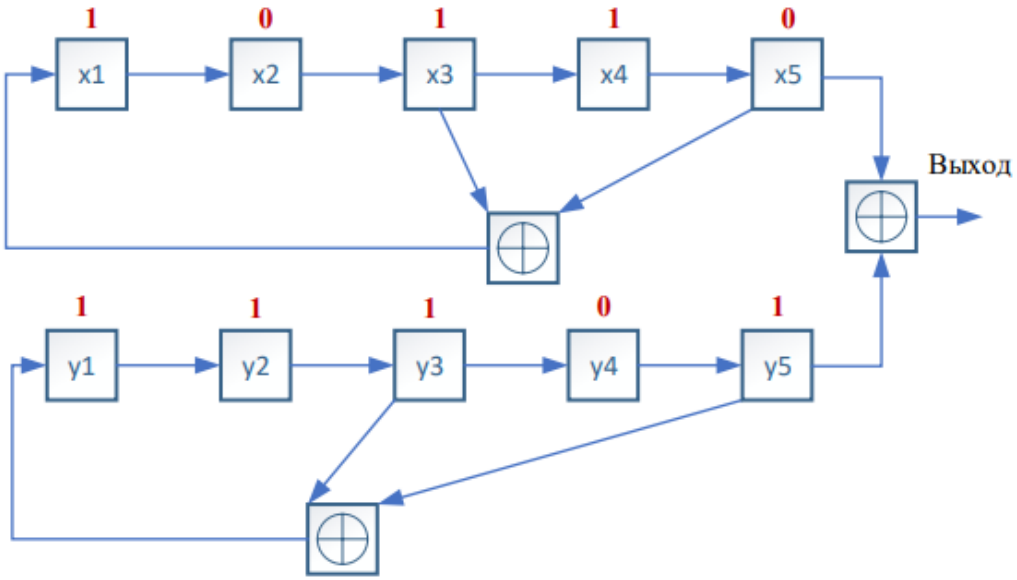


Рис. 4.4. Генерация последовательности Голда (вариант для *четного* номера группы)

```
1. int main(){
2. // Две последовательности битов  (линейно свдиговые регистры)
3. size_t m = 5;
4. int16_t Rex_x[] = {1, 0, 1, 0 , 0};
5. int16_t Rex_y[] = {1, 1, 0, 1 , 1};


6. size_t N = 31;
7. int16_t *out = (int16_t*)malloc(N * sizeof(int16_t));


8. for (size_t i = 0; i < N; i ++){
      a. out[i] = Rex_x[4] ^ Rex_y[4];


      b. temp1 = Rex_x[2] ^ Rex_x[4];
      c. temp2 = Rex_y[2] ^ Rex_y[4];


      d. for (size_t j = m-1; j > 0; j --){
            i.  Rex_x[j] = Rex_x[j-1];
           ii.  Rex_y[j] = Rex_y[j-1];
      e. }
      f. Rex_x[0] = temp1;
      g. Rex_y[0] = temp2;
9. }
```

```
show_arr("Rex_x[]", Rex_x, m);
show_arr("Rex_y[]", Rex_y, m);


show_arr("out[]", out, N);
```

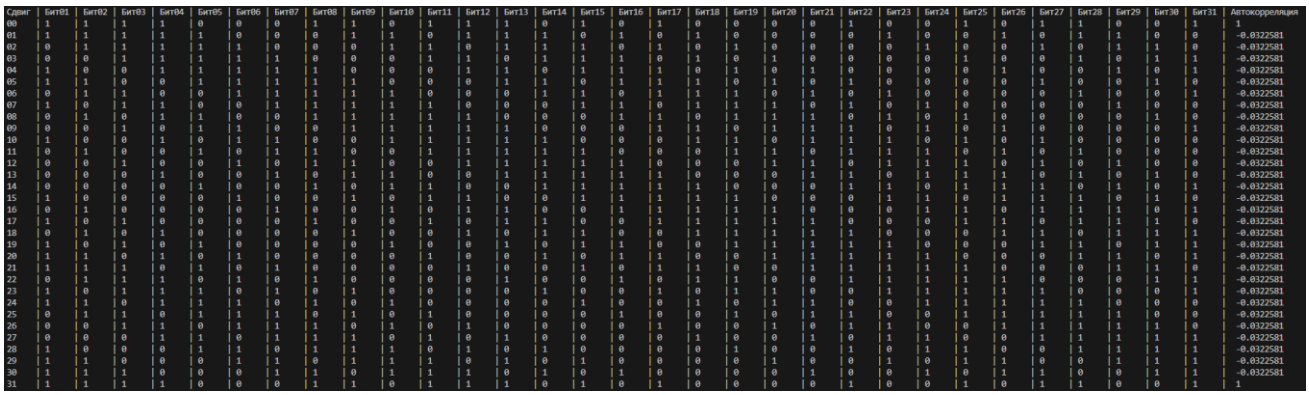
2. Выведите получившуюся последовательность на экран.

```
Rex_x[]: 10100
Rex_y[]: 11011
out[]: 11110001101110101000010010110001
```

3. Сделайте поэлементный циклический сдвиг последовательности и посчитайте автокорреляцию исходной последовательности и сдвинутой. Сформируйте таблицу с битовыми значениями последовательностей, в последнем столбце которой будет вычисленное значение автокорреляци

```
template <typename T>
void autocorr(const T *eq_array, size_t len){
double result[len];
cout << "Сдвиг | ";
for (size_t i = 1; i <= len; i ++){
cout << "Бит" << setw(2) << setfill('0') << i << " | ";
}
cout << "Автокорреляция" << endl;
T *array = (T*)malloc(len * sizeof(T));
for (size_t j = 0; j < len+1; j ++){
for (size_t i = 0; i < len; i ++){
array[i] = eq_array[i];
}
cout << setw(2) << setfill('0') << j << "    | ";
T *temp_shift = (T*)malloc(len * sizeof(T));
for (size_t i = 0; i < len; i++) {
temp_shift[i] = array[(i - j + len) % len];
}
for (size_t i = 0; i < len; i++) {
array[i] = temp_shift[i];
}
free(temp_shift);
for (size_t i = 0; i < len; i ++){
cout << array[i] << "     | ";
}
int16_t eq = 0;
int16_t neq = 0;
for (size_t i = 0; i < len; i ++){
if (array[i] == eq_array[i]){
eq ++;
} else {
neq ++;
}
```

```
34. }
35. result[j] = ((1/(double)len)*(eq-neq));
36. cout << " " << result[j] << endl;
37. }
38. free(array);
39. }
```



Сдвиг	Бит01	Бит02	Бит03	Бит04	Бит05	Бит06	Бит07	Бит08	Бит09	Бит10	Бит11	Бит12	Бит13	Бит14	Бит15	Бит16	Бит17	Бит18	Бит19	Бит20	Бит21	Бит22	Бит23	Бит24	Бит25	Бит26	Бит27	Бит28	Бит29	Бит30	Бит31	Автокорреляция
00	1	1	1	0	0	0	1	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
01	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
02	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
03	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
04	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
05	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
06	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
07	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
08	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
09	0	0	1	0	1	1	0	0	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	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
11	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
12	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
13	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
14	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
15	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
16	0	1	0	0	0	0	1	0	0	1	0	1	1	0	0	1	1	1	1	0	0	0	1	1	0	1	1	1	0	1	0	-0.0322581
17	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	1	0	1	1	1	-0.0322581
18	1	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
19	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
20	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
21	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	0	0	0	1	1	0	-0.0322581
22	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
23	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
24	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
25	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
26	0	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	1	0	1	1	1	0	0	1	1	1	1	1	0	-0.0322581
27	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
28	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
29	1	1	0	0	0	1	1	0	1	1	1	0	1	0	0	1	0	0	0	1	0	0	1	0	1	1	0	0	1	1	1	-0.0322581
30	1	1	1	0	0	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	1	0	1	1	0	0	1	1	0	-0.0322581
31	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	1

4. Сформируйте еще одну последовательность Голда, используя свою схему (рис.4.4 или 4.5), такую что x=x+1, а y= y-5. 4), вычислите значение взаимной корреляции исходной и новой последовательностей и выведите в терминал.

```
Rex_x[]: 10100
Rex_y[]: 11011
out[]: 11110001101110101000001001011001
Rex_x_2[]: 10101
Rex_y_2[]: 10110
out_2[]: 11000110111010100001001011001111
Корреляция исходной и новой битовой последовательности: -0.806452
tenebre@rxmantic:/mnt/e/prog/Third/MobileDr/1/4lab$
```

5. Проделайте шаги 1-5 в Matlab. Используйте функции xcorr() и autocorr() для вычисления соответствующих корреляций. Сравните результаты, полученные в Matlab и C/C++.

```
1.  clc; close all;
2.
3.  Rex_x = [1, 0, 1, 0, 0]
4.  Rex_y = [1, 1, 0, 1 , 1]
5.
6.  m = length(Rex_y);
7.  N = 31;
8.
9.  out = zeros(1, N);
10.
11. for i = 1:N
12.     out(i) = Rex_x(5) ~= Rex_y(5);
13.
14.     temp1 = Rex_x(3) ~= Rex_x(5);
15.     temp2 = Rex_y(3) ~= Rex_y(5);
16.
17.     for j = m:-1:2
18.         Rex_x(j) = Rex_x(j-1);
19.         Rex_y(j) = Rex_y(j-1);
20.     end
```

```
    Rex_x(1) = temp1;
    Rex_y(1) = temp2;
end

out

disp('Автокорреляция out')
disp(autocorr(out, N-1));

Rex_x_2 = [1, 0, 1, 0 , 1]
Rex_y_2 = [1, 0, 1, 1 , 0]

out_2 = zeros(1, N);

for i = 1:N
    out_2(i) = Rex_x_2(5) ~= Rex_y_2(5);

    temp1 = Rex_x_2(3) ~= Rex_x_2(5);
    temp2 = Rex_y_2(3) ~= Rex_y_2(5);

    for j = m:-1:2
        Rex_x_2(j) = Rex_x_2(j-1);
        Rex_y_2(j) = Rex_y_2(j-1);
    end
    Rex_x_2(1) = temp1;
    Rex_y_2(1) = temp2;
end

out_2

disp('Корреляция out и out2')
disp(xcorr(out, out_2))
```

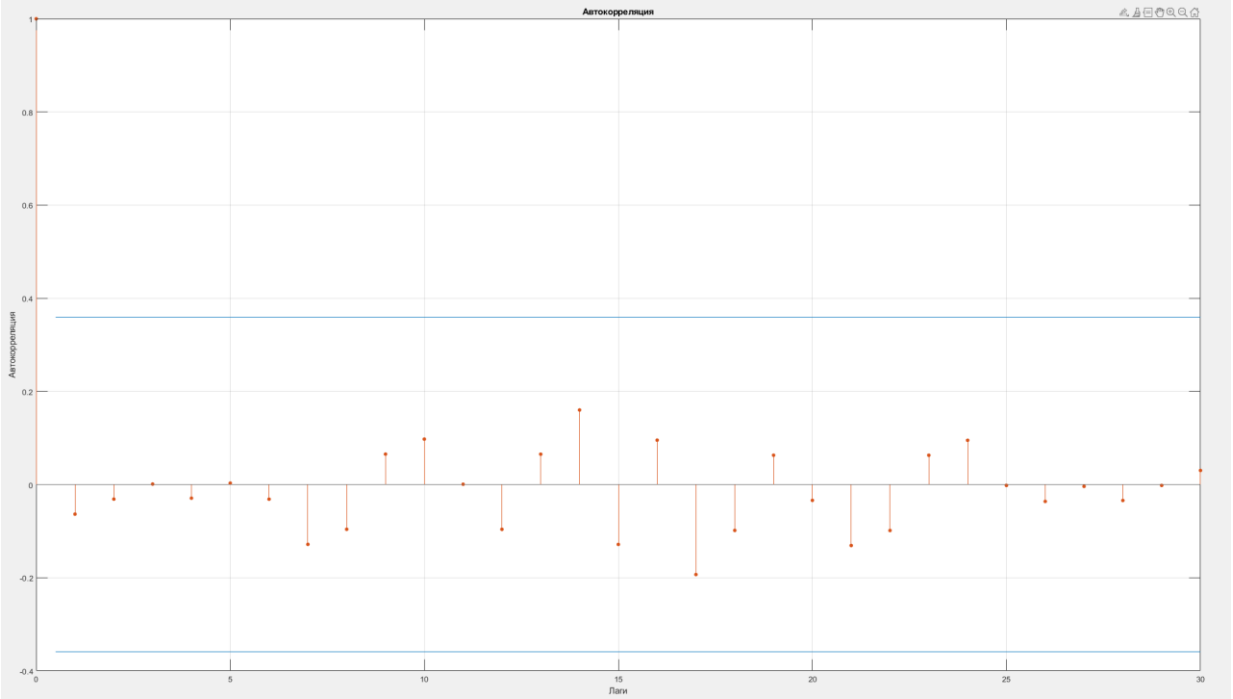
```
figure;
autocorr(out, N-1);
title('Автокорреляция');
xlabel('Лаги');
ylabel('Автокорреляция');
```

```
Rex_x =

     1     0     1     0     0


Rex_y =

     1     1     0     1     1


out =

  Columns 1 through 20

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

  Columns 21 through 40

     0     1     0     0     1     0     1     1     0     0     1


Автокорреляция out
  Columns 1 through 20

    1.0000   -0.0636   -0.0313    0.0009   -0.0293    0.0030   -0.0315   -0.1284   -0.0961    0.0653    0.0976    0.0007   -0.0962    0.0652    0.1599   -0.1286    0.0953   -0.1933   -0.0985    0.0629

  Columns 21 through 31

   -0.0340   -0.1309   -0.0987    0.0628    0.0950   -0.0019   -0.0363   -0.0040   -0.0343   -0.0020    0.0302


Rex_x_2 =

     1     0     1     0     1


Rex_y_2 =

     1     0     1     1     0


out_2 =

  Columns 1 through 20

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

  Columns 21 through 40

     0     0     1     0     1     1     0     0     1     1     1


Корреляция out и out2
  Columns 1 through 20

    1.0000    2.0000    3.0000    3.0000    2.0000    2.0000    2.0000    3.0000    5.0000    4.0000    3.0000    4.0000    5.0000    5.0000    4.0000    3.0000    6.0000    4.0000    6.0000    5.0000

  Columns 21 through 40

    4.0000    6.0000    7.0000    6.0000    6.0000    6.0000    7.0000    8.0000    7.0000    7.0000    8.0000    7.0000   14.0000    5.0000    5.0000    6.0000    6.0000    6.0000    5.0000    3.0000

  Columns 41 through 60

    4.0000    5.0000    4.0000    3.0000    3.0000    4.0000    5.0000    2.0000    4.0000    2.0000    3.0000    4.0000    2.0000    1.0000    2.0000    2.0000    2.0000    1.0000         0    1.0000

  Column 61

    1.0000
```

Результаты так отличаются потому что в си++ мы использовали определенную формулу, у которой нет центрирования, нормировка по длине битов.

А в функциями матлаб, в autocorr например имеется центрирование и нормировка происходит по дисперсии, в xcorr же нормировки в общем нет.

6. Выведите на график в Matlab функцию автокорреляции в зависимости от величины задержки (lag).



Контрольные вопросы:

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

- 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 радио условиях, 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 сравнении двух сигналов (или последовательностей) они совпадают чаще, чем различаются.

3) Что такое корреляционный прием сигналов?

- Корреляционный приём — это метод обнаружения известного сигнала на фоне шума, при котором принимаемый сигнал умножается (коррелируется) с заранее известной копией (эталоном), и результат интегрируется за период символа.
  Если на входе присутствует сигнал, совпадающий с эталоном, значение корреляции достигает максимума (пика).

4) Как вычисление корреляционных функций помогает синхронизироваться приемникам и передатчика в сетях мобильной связи?

- При включении абонентского терминала он не знает, где начинаются слоты или кадры в принимаемом сигнале.
  Базовая станция периодически передаёт известную синхронизирующую PN-последовательность.
  Терминал вычисляет корреляцию между принятым сигналом и этой эталонной последовательностью при различных временных сдвигах.
  В момент, когда достигается максимальный корреляционный пик, приёмник

определяет точное начало временного слота и устанавливает временную синхронизацию с передатчиком.

5) Какими свойствами обладают псевдослучайные битовые последовательности?

1. Сбалансированность: количество единиц и нулей на периоде отличается не более чем на 1 (например, 16 единиц и 15 нулей при длине 31).
2. Цикличность: распределение циклов (серий одинаковых битов) подчиняется закону
3. Хорошие корреляционные свойства:
    - автокорреляция при нулевом сдвиге = +1,
    - при любом другом сдвиге — близка к нулю (для m-последовательностей: $-1/p$).

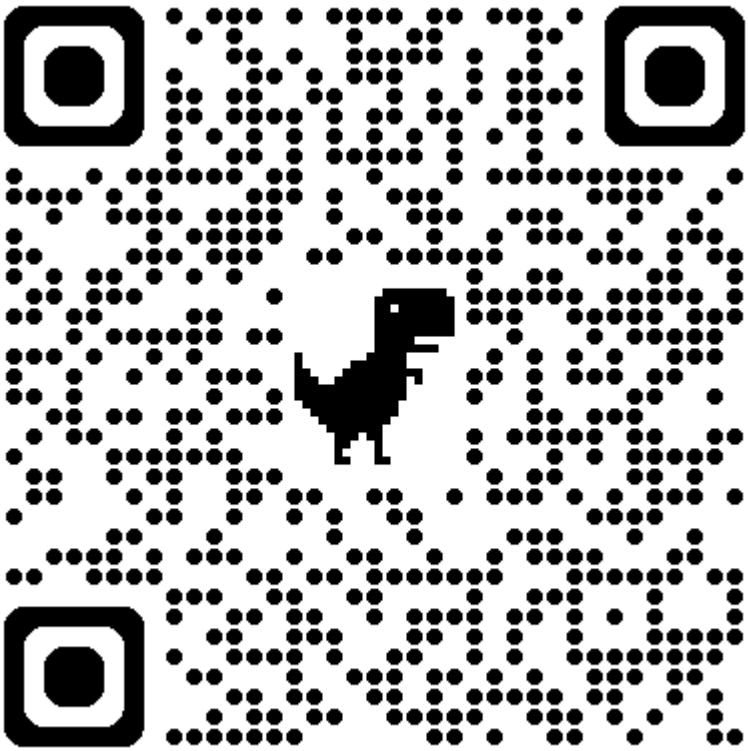
6) Какие разновидности PN-последовательностей вам известны?

- m-последовательности (maximum-length sequences) — генерируются LFSR, обладают хорошей автокорреляцией.
- Коды Голда — формируются XOR двух m-последовательностей одинаковой длины; обладают хорошими взаимокорреляционными свойствами.
- Коды Касами — также основаны на m-последовательностях, используют периодическую выборку и суммирование; имеют низкую взаимную корреляцию.
- Коды Баркера — короткие последовательности с идеальной автокорреляцией (боковые лепестки = 0), но малой длиной.
- Коды Уолша–Адамара — ортогональные последовательности, используются в CDMA для разделения каналов.

Вывод

В ходе выполнения работы были сгенерированы две последовательности Голда на основе заданных начальных состояний LFSR, изучены их автокорреляционные и взаимокорреляционные свойства. Показано, что последовательности Голда обладают выраженным автокорреляционным пиком при нулевом сдвиге и низким уровнем боковых лепестков при других сдвигах, что подтверждает их пригодность для задач синхронизации в мобильных сетях.

Гитхаб



https://github.com/ohfuckinglucy/OSMS