Министерство цифрового развития ФГБОУ ВО “СибГУТИ”

Основы систем мобильной связи

Отчет
Практическое занятие 4
Вариант 1

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

Выполнил студент группы ИА-231
Готфрид Матвей

г. Новосибирск

1. 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.

Краткая теория
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 PNпоследовательность 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
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.
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 синхроимпульсами 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.

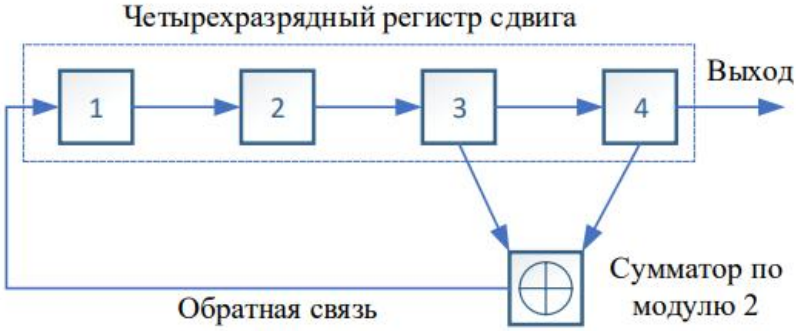


Рис. 4.1. Пример способа формирования псевдослучайной битовой последовательности.

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 T0 в нормированной форме (4.1) - (4.2)

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

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

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

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

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

---

o c c o o c o c o o o o c c c

o – отличаются; c – совпадают. Число совпадений: 7; Число несовпадений: 8. Следовательно,

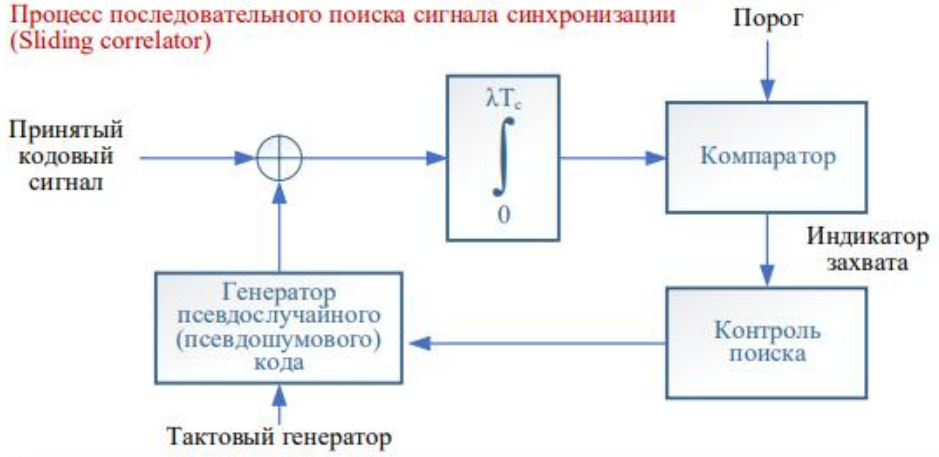
$$R_x(\tau = 1) = \frac{1}{15}(7 - 8) = -\frac{1}{15}.$$

Разновидности псевдо-шумовых битовых последовательностей

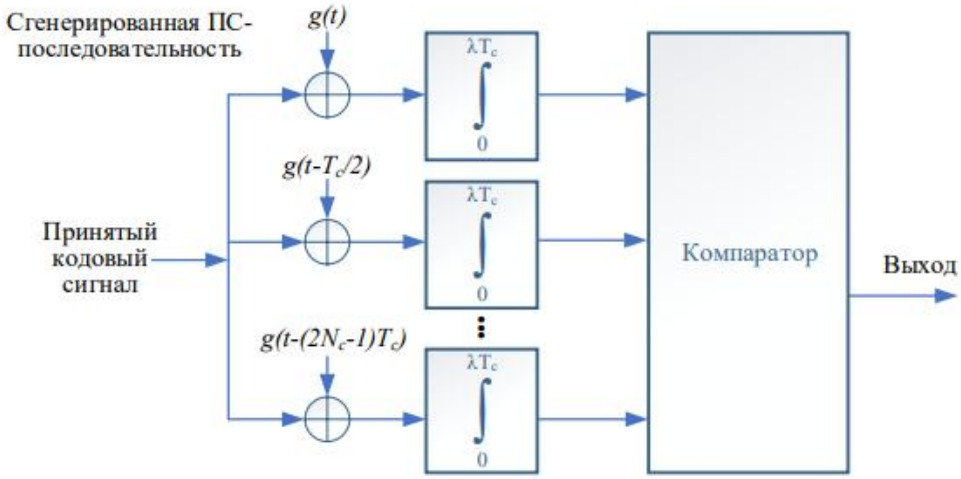
М-последовательности – не единственные PN-последовательности, используемые в системах мобильной связи. Существуют также коды Баркера, коды Голда, коды Касами, коды Уолша-Адамара.

Коды Голда формируются путем суммирования по модулю 2 двух М-последовательностей одинаковой длины. Коды Касами также формируются из М-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



Выполнение работы.

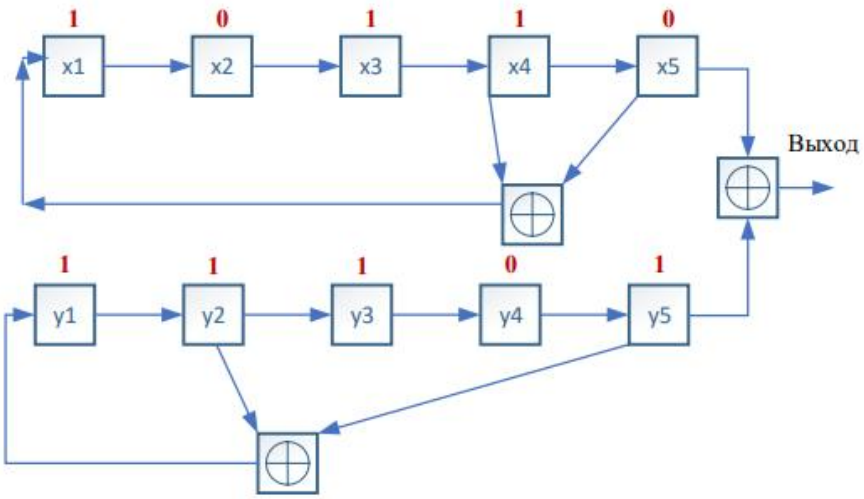


Рис. 4.5. Генерация последовательности Голда (вариант для *нечетного* номера группы)

```c
#include <stdio.h>
#include <math.h>

#define N 5
#define SEQ_LENGTH ((1 << N) - 1)
#define SUM(x, y) ((x + y) % 2)
#define TOLERANCE 0.5 // Допустимая погрешность для автокорреляции
double autocorr(int x, int y, int length)
{
    int parity = 0;
    int disparity = 0;
    for (int i = 0; i < length; i++)
    {
        if (((x >> i) & 1) == ((y >> i) & 1))
            parity++;
        else
            disparity++;
    }
    return (double)(parity - disparity) / length;
}
int Gold_sequence(int x, int y)
{
    int res = 0;
    for (int i = 0; i < SEQ_LENGTH; i++)
    {
        int x_feedback = SUM(((x >> 3) & 1), (x & 2));
        int y_feedback = SUM(((y >> 2) & 1), (y & 3));
        res = (res >> 1) | (SUM((x & 1), (y & 1)) << (SEQ_LENGTH - 1));
        x = (x >> 1) | (x_feedback << (N - 1));
        y = (y >> 1) | (y_feedback << (N - 1));
    }
    return res;
}
void print_binary(int x, int length)
{
    for (int i = length - 1; i >= 0; i--)
```

```
        if (x & (1 << i))
            printf("1");
        else
            printf("0");
    printf("\t");
}
int shift_seq(int x, int length)
{
    int bit = (x & 1) << (length - 1);
    return (bit | (x >> 1)) & ((1 << length) - 1);
}
int count_ones(int sequence, int length)
{
    int count = 0;
    for (int i = 0; i < length; i++)
    {
        if ((sequence >> i) & 1)
            count++;
    }
    return count;
}
// Проверка сбалансированности
int check_balance(int sequence, int length)
{
    int ones = count_ones(sequence, length);
    if (abs(ones - (length - ones)) > 1)
    {
        printf("Сбалансированность не выполнена: 1=%d, 0=%d\n", ones, length - ones);
        return 0;
    }
    return 1;
}
int check_cycle_distribution(int seq, int length)
{
    int current_length = 1;
    int last_bit = seq & 1;
    int counts[length];
    for (int i = 0; i < length; i++)
        counts[i] = 0;
    for (int i = 1; i < length; i++)
    {
        int bit = (seq >> i) & 1;
        if (bit == last_bit)
        {
            current_length++;
        }
        else
        {
            if (current_length < length)
                counts[current_length]++;
            current_length = 1;
            last_bit = bit;
        }
    }
    for (int i = 1; i < length; i++)
    {
        if (counts[i] > 0)
            printf("Циклы длиной %d встречаются %d раз(а)\n", i, counts[i]);
    }
    return 1;
}
```

```c
int check_autocorrelation(int sequence, int length)
{
    for (int shift = 0; shift < length; shift++)
    {
        int shifted_sequence = shift_seq(sequence, length);
        double correlation = autocorr(sequence, shifted_sequence, length);
        if (shift != 0 && fabs(correlation) > TOLERANCE)
        {
            printf("Автокорреляция на сдвиге %d не близка к 0: %lf\n", shift, correlation);
            return 0;
        }
    }
    printf("Автокорреляция соблюдена!\n");
    return 1;
}
// Основная функция проверки
int is_m_sequence(int sequence, int length)
{
    int is_valid = 1;
    printf("Проверка сбалансированности:\n");
    if (!check_balance(sequence, length))
        is_valid = 0;
    printf("Проверка распределения циклов:\n");
    if (!check_cycle_distribution(sequence, length))
        is_valid = 0;
    printf("Проверка автокорреляции:\n");
    if (!check_autocorrelation(sequence, length))
        is_valid = 0;
    return is_valid;
}
int main()
{
    int x = 0b10010;
    int y = 0b01010;
    int seq = Gold_sequence(x, y);
    if (is_m_sequence(seq, SEQ_LENGTH))
    {
        printf("Это М последовательность\n");
    }
    else
    {
        printf("Это не М последовательсть\n");
    }
    int new_seq = seq;
    printf("Сдвиг\tПоследовательность Голда\tАвтокорреляция\n");
    printf("%d\t", 0);
    print_binary(seq, SEQ_LENGTH);
    printf("%lf\n", autocorr(seq, new_seq, SEQ_LENGTH));
    for (int i = 1; i < SEQ_LENGTH + 1; i++)
    {
        printf("%d\t", i);
        new_seq = shift_seq(new_seq, SEQ_LENGTH);
        print_binary(new_seq, SEQ_LENGTH);
        printf("%lf\n", autocorr(seq, new_seq, SEQ_LENGTH));
    }
    printf("\n");
    x += 1;
    y -= 5;
    seq = Gold_sequence(x, y);
    if (is_m_sequence(seq, SEQ_LENGTH))
    {
```

```
        printf("Это M последовательность\n");
    }
    else
    {
        printf("Это не M последовательность\n");
    }
    new_seq = seq;
    printf("Сдвиг\tПоследовательность Голда\tАвтокорреляция\n");
    printf("%d\t", 0);
    print_binary(seq, SEQ_LENGTH);
    printf("%lf\n", autocorr(seq, new_seq, SEQ_LENGTH));
    for (int i = 1; i < SEQ_LENGTH + 1; i++)
    {
        printf("%d\t", i);
        new_seq = shift_seq(new_seq, SEQ_LENGTH);
        print_binary(new_seq, SEQ_LENGTH);
        printf("%lf\n", autocorr(seq, new_seq, SEQ_LENGTH));
    }
    return 0;
}
```

X = 18
Y = 10

```
Проверка сбалансированности:
Проверка распределения циклов:
Циклы длиной 1 встречаются 7 раз(а)
Циклы длиной 2 встречаются 5 раз(а)
Циклы длиной 3 встречаются 3 раз(а)
Циклы длиной 4 встречаются 1 раз(а)
Проверка автокорреляции:
Автокорреляция соблюдена!
Это M последовательность
```

Сдвиг Последовательность Голда Автокорреляция

0 01000110110101100111100010110001.000000
1 00100011011010110011110001011000-0.032258
2 00010001101101011001111000101100-0.161290
3 00001000110110101100111100010110-0.032258
4 10000100011011010110011110001010-0.161290
5 01000010001101101011001111000101 0.096774
6 10100001000110110101100111100010 0.096774
7 01010000100011011010110011110001-0.032258
8 10101000010001101101011001111000 0.096774
9 01010100001000110110101100111100-0.032258
10 00101010000100011011010110011110-0.032258
11 10010101000010001101101011001111-0.290323
12 11001010100001000110110101100110.096774
13 11100101011000010001101101011001 0.354839

14 11110001011000010001101101011000 -0.290323
15 01111000101100001000110110101100 -0.161290
16 00111100010110000100011011010110 -0.161290
17 10011110001011000010001101101011 -0.290323
18 11001111000101100001000110110101 0.354839
19 01100111100010110000100011011010 0.096774
20 10110011110001011000010001101101 -0.290323
21 01011001111000101100001000110110 -0.032258
22 10101100111100010110000100011011 -0.032258
23 11010110011110001011000010001101 0.096774
24 01101011001111000101100001000110 -0.032258
25 10110101100111100010110000100011 0.096774
26 11011010110011110001011000010001 0.096774
27 01101101011001111000101100001000 -0.161290
28 00110110101100111100010110000100 -0.032258
29 00011011010110011110001011000010 -0.161290
30 10001101101011001111000101100001 -0.032258
31 01000110110101100111100010110000 1.000000

```
Проверка сбалансированности:
Проверка распределения циклов:
Циклы длиной 1 встречаются 9 раз(а)
Циклы длиной 2 встречаются 5 раз(а)
Циклы длиной 3 встречаются 2 раз(а)
Циклы длиной 4 встречаются 1 раз(а)
Проверка автокорреляции:
Автокорреляция соблюдена!
Это M последовательность
```

Сдвиг Последовательность Голда Автокорреляция
0 11011100100101001100001110101101 1.000000
1 01101110010010100110000111010110 -0.161290
2 10110111001001010011000011101011 -0.161290
3 11011011100100101001100001110101 0.096774
4 01101101110010010100110000111010 -0.161290
5 10110110111001001010011000011101 0.096774
6 01011011011100100101001100000111 0.096774
7 10101101101110010010100110000011 -0.032258
8 11010110110111001001010011000011 0.096774
9 11101011011011100100101001100001 -0.161290
10 01110101101101110010010100110000 -0.032258
11 00111010110110111001001010011000 -0.032258

12 000111010110110111001001010011 0.096774
13 000011101011011011100100101011 0.225806
14 100001110101101101110010010100 1 -0.419355
15 110000111010110110111001001010 0 -0.032258
16 011000011101011011011100100101 0 -0.032258
17 001100001110101101101110010010 1 -0.419355
18 100110000111010110110111001001 0 0.225806
19 010011000011101011011011100100 1 0.096774
20 101001100001110101101101110010 0 -0.032258
21 010100110000111010110110111001 0 -0.032258
22 001010011000011101011011011100 1 -0.161290
23 100101001100001110101101101110 0 0.096774
24 010010100110000111010110110111 0 -0.032258
25 001001010011000011101011011011 1 0.096774
26 100100101001100001110101101101 1 0.096774
27 110010010100110000111010110110 1 -0.161290
28 111001001010011000011101011011 0 0.096774
29 011100100101001100001110101101 1 -0.161290
30 101110010010100110000111010110 1 -0.161290
31 110111001001010011000011101011 0 1.000000

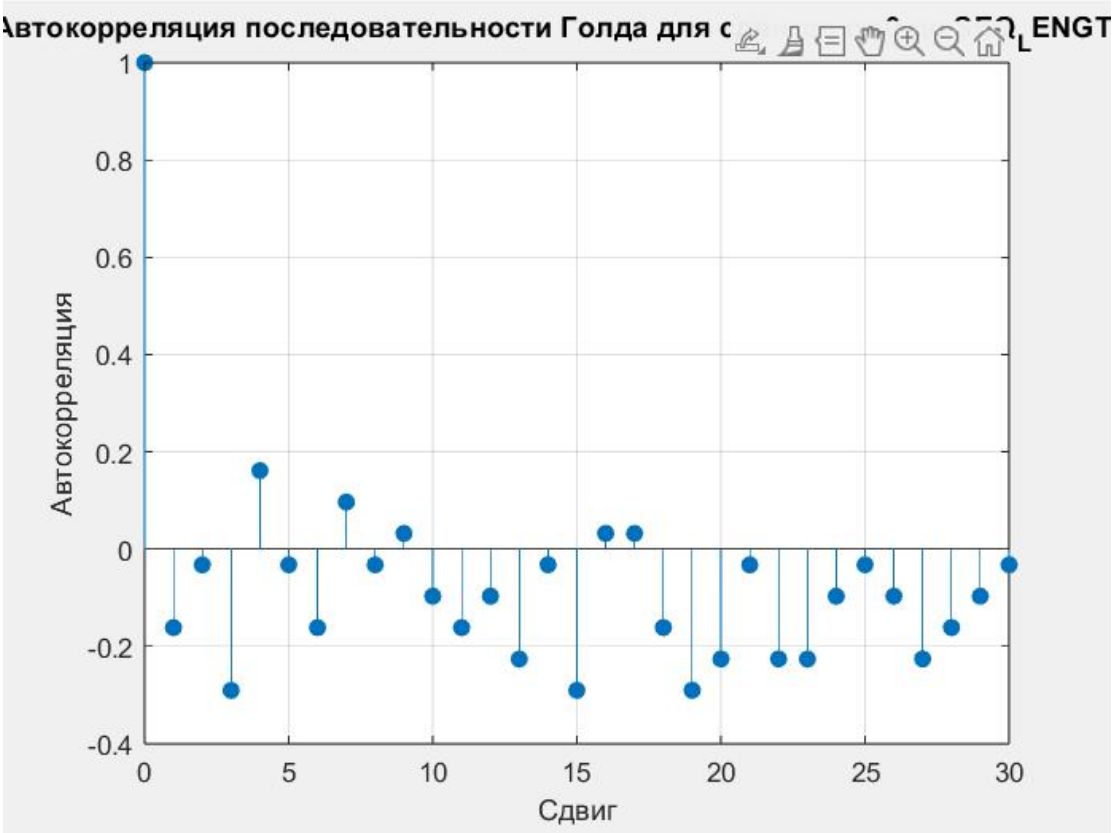
X = 2
Y = 3

```
N = 5;
SEQ_LENGTH = 2^N - 1;

x = 18;  % 0b10010
y = 10;  % 0b01010
```

```
% Функция вычисления автокорреляции
function correlation = autocorr(x, y, length)
parity = 0;
disparity = 0;
for i = 1:length
    if bitget(x, i) == bitget(y, i)
        parity = parity + 1;
    else
        disparity = disparity + 1;
    end
end
correlation = (parity - disparity) / length;
end
```

```
% Функция генерации последовательности Голда
function res = Gold_sequence(x, y, N, SEQ_LENGTH)
res = 0;
for i = 1:SEQ_LENGTH
    x_feedback = mod(bitget(x, 4) + bitget(x, 2), 2);
```

```
    y_feedback = mod(bitget(y, 3) + bitget(y, 2), 2);

    res = bitset(res, SEQ_LENGTH - i + 1, mod(bitget(x, 1) + bitget(y, 1), 2));

    x = bitshift(x, -1) + (x_feedback * 2^(N - 1));
    y = bitshift(y, -1) + (y_feedback * 2^(N - 1));
end
end

function shifted = shift_seq(x, length)
shifted = bitshift(x, -1);
shifted = bitor(shifted, bitget(x, 1) * 2^(length - 1));
end

gold_seq = Gold_sequence(x, y, N, SEQ_LENGTH);

auto_corr_values = zeros(1, SEQ_LENGTH);

for shift = 0:SEQ_LENGTH-1
    shifted_seq = bitshift(gold_seq, shift);
    auto_corr_values(shift + 1) = autocorr(gold_seq, shifted_seq, SEQ_LENGTH);
end

disp('Корреляции со сдвигами от 0 до SEQ_LENGTH-1:');
disp(auto_corr_values);

% Визуализация автокорреляции
figure;
stem(0:SEQ_LENGTH-1, auto_corr_values, 'filled');
title('Автокорреляция последовательности Голда для сдвигов от 0 до SEQ_LENGTH-1');
xlabel('Сдвиг');
ylabel('Автокорреляция');
grid on;
```



Контрольные вопросы

1. Псевдослучайные последовательности (PN-последовательности) используются для кодирования сигнала в мобильных сетях, чтобы уменьшить помехи и обеспечить эффективную передачу данных. Они помогают в разделении пользователей в сетях с множественным доступом (например, CDMA) и обеспечивают устойчивость связи в условиях сильных помех, выполняя функции шифрования, синхронизации и мультиплексирования.
2. Положительная корреляция сигналов означает, что сигналы имеют схожие формы или совпадают по фазе в некоторых точках во времени. Высокая положительная корреляция показывает, что сигналы сонаправлены, что используется, например, для обнаружения похожих сигналов и их синхронизации в приемниках.
3. Корреляционный прием сигналов — это метод обработки, при котором принимаемый сигнал сравнивается с эталонной копией сигнала (например, известной PN-последовательностью) для определения степени их схожести. Этот метод позволяет выделить нужный сигнал из шума и других помех и используется для синхронизации и демодуляции сигналов в мобильных системах связи.
4. Корреляционный прием сигналов — это метод обработки, при котором принимаемый сигнал сравнивается с эталонной копией

сигнала (например, известной PN-последовательностью) для определения степени их схожести. Этот метод позволяет выделить нужный сигнал из шума и других помех и используется для синхронизации и демодуляции сигналов в мобильных системах связи.

5. Псевдослучайные битовые последовательности обладают следующими свойствами:

- Автокорреляция: высокая автокорреляция на нулевом сдвиге и близкая к нулю на всех других сдвигах, что позволяет легко выделять сигналы среди помех.
- Периодичность: последовательности повторяются через определенное количество бит, но выглядят случайными.
- Спектральная равномерность: имеют спектральные свойства, близкие к белому шуму, что помогает в маскировке сигнала и снижении перекрестных помех.
- Детерминированность: могут воспроизводиться при тех же начальных условиях, что упрощает синхронизацию и воспроизведение сигналов.

6. Существуют различные разновидности PN-последовательностей, среди которых:

- Последовательности Голда: создаются на основе двух m-последовательностей, имеют хорошую автокорреляцию и подходят для CDMA.
- Последовательности Касперов: также используются в CDMA для уменьшения помех и повышения надежности передачи.
- m-последовательности: генерируются с использованием линейных регистров сдвига с обратной связью, обладают хорошей автокорреляцией и равномерным распределением единиц и нулей.
- Последовательности Баркера: короткие последовательности с отличной автокорреляцией, часто используются в радарных системах.