Лабораторная работа № 1

Методы кодирования и модуляции сигналов.

Цели работы

Изучение методов кодирования и модуляции сигналов в телекоммуникационных сетях с помощью пакета компьютерного моделирования Matlab. Определение спектра и параметров сигнала. Демонстрация принципов модуляции сигнала на примере аналоговой амплитудной модуляции. Исследование свойства самосинхронизации сигнала.

Теоретические сведения

1. Основные работы в Matlab.

Matlab – высокоуровневый интерпретируемый язык программирования, предназначенный для решения задач вычислительной математики.

В окне интерпретатора Matlab пользователь может вводить как отдельные команды языка Matlab, так и группы команд, объединяемые в программы. Если строка заканчивается символом «;», то результаты на экран не выводятся. Если в конце строки символ «;» отсутствует, то результаты работы выводятся на экран. Текст в строке, который идет после символа % является строкой комментария и интерпретатором не обрабатывается.

Matlab имеет два режима работы: *терминальный* и *программный*. В терминальном режиме отдельные команды последовательно вводятся в окне интерпретатора. В программном режиме создается текстовый файл (с решением .m), в котором хранятся последовательно выполняемые команды, впоследствии запускаемые на выполнение в среде Matlab.

Простейшие арифметические операции в Matlab:

+ - сложение;

– - вычитание;

***** - умножение;

/ - деление;

^ - возведение в степень.

Для определения переменной необходимо набрать *имя переменной*, символ «=» и *значение переменной*, где знак равенства – это оператор присваивания:

имя переменной = значение выражения

Система различает большие и малые буквы в именах переменных. Выражение в правой части оператора присваивания может быть числом, арифметическим выражением, строкой символов или символьным выражением. Если речь идет о символьной или строковой переменной, то выражение в правой части оператора присваивания следует брать в одинарные кавычки.

Если команда не содержит знака присваивания, то по умолчанию вычисленное значение присваивается специальной системой переменной *ans.* Причем полученные значения можно использовать в последующих вычислениях, но важно помнить, что значение *ans* изменяется после кажого вызова команды без оператора присваивания.

Системные применения:

- *ans* – результат последней операции без знака присваивания;
- *i, j* – мнимая единица ($\sqrt{-1}$);
- *pi* – число π (3, 141592653589793);
- *e* – число e (экспонента 2,71828183);
- *inf* – машинный символ бесконечности (∞);
- *NaN* – неопределенный результат.

Все перечисленные переменные можно использовать в математических выражениях.

Команда *clear* предназначена для уничтожения определения одной или нескольких переменных

clear имя переменной

В общем виде обращение к функции в Matlab имеет вид

имя переменной = имя функции (аргумент)

или

имя функции (аргумент)

Если имя переменной указано, то ей будет присвоен результат работы функции. Если же оно отсутствует, то значение вычисленного функцией результата присваивается системной переменной *ans.* Примеры работы с тригонометрическими функциями

```
>>x=pi/2;        % Определение значения аргумента
>>y=sin(x)       % Вызов функции
y=1
>>cos(pi/3)      % Вызов функции
```

ans=0.50000

Здесь >> - знак приглашения Matlab для ввода команд. Некоторые встроенные функции Matlab:

- *sin(x)* – синус числа *x*;
- *cos(x)* – косинус числа *x*;
- *tan(x)* – тангенс числа *x*;
- *exp(x)* – экспонента числа *x*;
- *log(x)* – натуральный логарифм числа *x*;
- *round(x)* – обычное округление числа *x* до ближайшего целого;
- *mod (x,y)* – вычисление остатка от деления *x* на *y*;
- *sign (x)* – сигнум-функция числа *x*, выдает 0, если = 0, -1 – при *x* < 0 и 1 при *x* > 0;
- *sqrt (x)* – корень квадратный из числа *x*;
- *abs (x)* – модуль числа *x*;

Операции отношения выполняют сравнение двух операндов и определяют, истинно выражение или ложно:

< - меньше;
> - больше;
= = - равно;
~= - не равно;
<= - меньше или равно;
>= - больше или равно.
Синтаксис функции определяемой пользователем:
function name1 [,namel2,...]= fun (var1 [,var2, ...])

Здесь *name [,name2, ...]* – список выходных параметров, т.е. переменных, которым будет присвоен конечный результат вычислений.
fun – имя функции, *var1 [, var2, ...]* – входные параметры.

Все имена переменных внутри функции, а также имена из списка входных и выходных параметров воспринимаются системой как локальные, т.е. эти переменные считаются определенными только внутри функции. Программы и функции в Matlab могут быть созданы при помощи текстового редактора и сохранены в виде файла с расширением *.m.* Но при создании и сохранении функции следует помнить, что ее имя должно совпадать с именем файла. программу можно запустит на выполнение, указав имя файла, в котором она сохранена. Обращение к функции осуществляется так же, как и к любой другой встроенной функции системы, т.е. с указанием входных и

выходных параметров. Вызывать функцию можно из командной строки или использовать ее как один из операторов программы.

Массив – множественный тип данных, состоящий из фиксированного числа элементов одного типа. Как и любой другой переменной, массиву должно быть присвоено имя.

Самый простой способ задать одномерный массив в Matlab имеет вид

имя массива = Xn:dX:Xk

Здесь *Xn* – значение первого элемента массива, *Xk* – значение последнего элемента массива, *dX* – шаг, с помощью которого формируется каждый следующий элемент массива, т.е. значение второго элемента составит

Xn+dX, третьего *Xn+dX+dX* и т.д. до *Xk*. Примеры создания массивов:

```
>> A=1:5
A=
1 2 3 4 5
>>B=2:2:10
B=
2 4 6 8 10
>>xn=3.5; xk=3.5; dx=0.5;
>>X=xn:dx:xk
X=
Columns 1 through 8:
-3.5 -3.0 -2.5 -2.0 -1.5 -1.0 -0.5 0.0
Columns 9 thought 15:
0.5 1.0 1.5 2.0 2.5 3.0 3.5
```

Обратиться к элементу вектора можно, указав имя массива и порядковый номер элемента в круглых скобах:

```
>> x= [2 4 6 8 10];
>> y= [-1.2 3.4 -0.8 9.1 5.6 -7.3];
>> x(1)  % значение первого элемента массива x
ans =2
>> y(5)  % значение пятого элемента массива y
ans = 5.6000
>> x(1)/2+y(3) -2-x(4)/y(5)
ans = 0.21143
```

Ввод элементов матрицы также осуществляется в квадратных скобках, при этом элементы строки отделяются друг от друга пробелом или запятой, а

строки разделяются между собой точкой с запятой. Обратиться к элементу матрицу можно, указав после имени матрицы, в круглых скобках, через запятую номер строки и номер столбца, на пересечении которых элемент расположен:

```
>>M=[2 4 6; 3 5; 7 8 9]
M=
2 4 6
1 3 5
7 8 9
>>M(1;2)
ans = 4
>>M(3;1)
ans = 7
```

Построение графиков в Matlab. Для того чтобы построить двумерный график *f(x)*, необходимо сформировать два массива *x* и *y* одинаковой размерности, а затем обратиться к функции *plot:*

Синтаксис функции *plot:*

plot(x1, y1, s1, x2, y2, s2,…, xn, yn, sn)

Здесь x1, x2,…,xn – массив абсцисс графиков; y1, y2,…,yn –массив ординат графиков; s1, s2, …, sn – строка форматов определяющая параметры линии и, при необходимости, позволяющая вывести легенду.

В строке форматов могут участвовать символы, отвечающие за тип линии, маркер и его размер, цвет линии и вывод легенды. За сплошную линию отвечает символ «-». Цвет линии определяется буквой латинского алфавита: y – желтый, m – розовый, c – голубой, r – красный, g – зеленый, b – синий, w – белый. Некоторые символы маркера: . – точка, * - звездочка, x – крестик, + - плюс, o – незакрашенный круг, p – незакрашенный квадрат.

Например, для построения графика функции *y= sin x + (1/3) sin 3x + (1/5) sin 5x* на интервале [-10; 10] (рис. Л.1) можно использовать следующий листинг:

```
clear all
close all
clc
% Формирования массива x:
x=-10:0.1:10;
% Формирование массива y.
y= sin(x) + 1/3*sin(3*x) + 1/5*sin(5*x);
% Построение графика функции:
figure(1)
plot (x,y, '-ok', 'markersize', 4)
```

```
% Отображение сетки на графике
grid on;
% Подпись оси X:
xlabel('x');
% Подпись оси Y:
ylabel('y');
% Название графика:
title ('y= sin(x) + 1/3*sin(3*x) + 1/5*sin(5*x)');
```

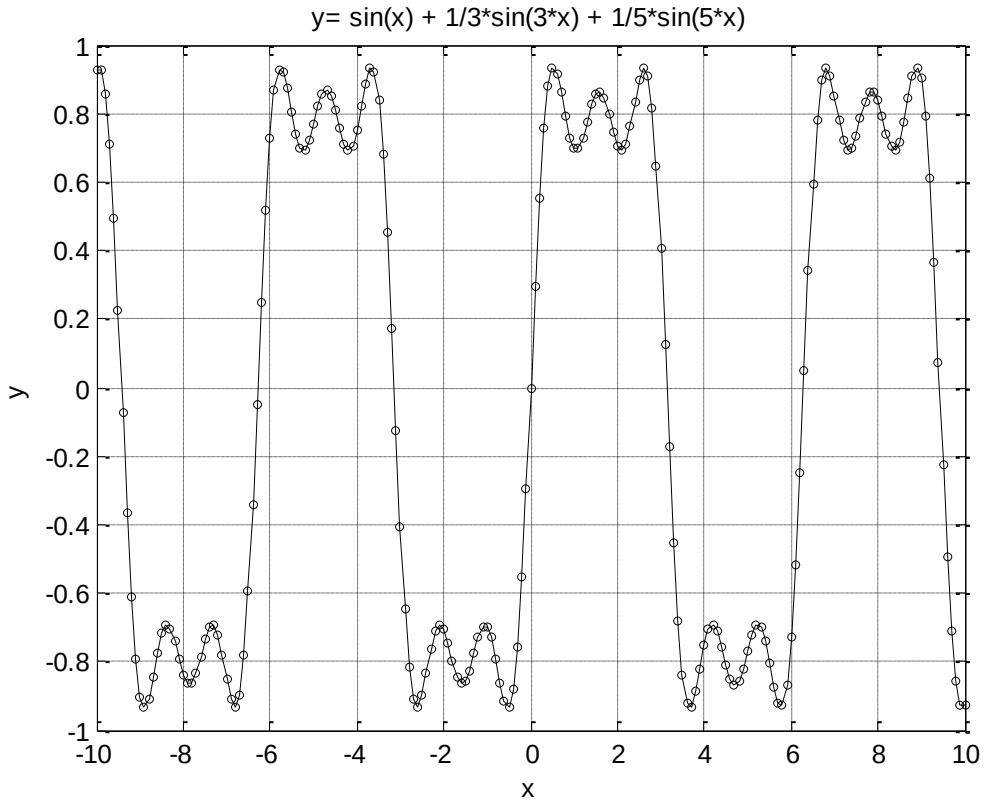

Рис. Л.1. **Графика функции *y= sin x + (1/3) sin 3x + (1/5) sin 5x* на интервале [-10; 10]**

Если повторно обратиться к функции *plot*, то в этом же окне будет стёрт первый график и нарисован второй. Для построения нескольких графиков в одной системе координат можно поступить одним из следующих способов:

1) обратиться к функции *plot* следующим образом:
plot(x1,y1,x2,y2,... xn,yn),

где *x1*, *y1* — массивы абсцисс и ординат первого графика, *x2*, *y2* — массивы абсцисс и ординат второго графика, ..., *xn*, *yn* — массивы абсцисс и ординат n-го графика;

2) каждый график изображать с помощью функции *plot(x,y),* но перед обращением к функциям *plot(x2,y2), plot(x3,y3), ..., plot(xn,yn)* вызвать команду *hold on,* которая блокирует режим очистки окна.

2. Модуляция сигналов

Модуляция – процесс изменения одного или нескольких параметров высокочастотного несущего колебания по закону низкочастотного информационного сигнала (сообщения).

Несущая – высокочастотное колебание, выполняющее роль переносчика информации, заложенной в управляющем (модулирующем) сигнале.

В качестве несущего колебания наиболее часто используются гармоническое колебание. В зависимости от того, какой из параметров несущего колебания - амплитуда, частота или начальная фаза несущего колебании изменяется по закону передаваемого сообщения, различают виды модуляции, соответственно, амплитудная, частотная или фазовая.

Сигнал, получаемый в процессе модуляции, называют модулированным колебанием, или радиосигналом.

Модуляция аналогового сигнала требуется в случае передачи низкочастотного (например, голосового) сигнала через канал высокочастотной области спектра амплитуда высокочастотного несущего сигнала изменяется (модулируется) в соответствии с изменением низкочастотного сигнала.

Методы преобразования передаваемого сигнала делятся на временные и частотные. Эквивалентность таких частотно-временных изменений обеспечивается однозначно через преобразование Фурье.

Преобразование Фурье – операция, сопоставляющая функции вещественной переменой другую функцию вещественной переменной, которая описывает коэффициенты (амплитуды) при разложении исходной функции на элементарные составляющие – гармонические колебания с разными частотами.

Для функции $f(x) \in R$ преобразование Фурье имеет вид

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(x) e^{-ix\omega} dx.$$

Обратное преобразование Фурье функции $f(\omega)$

$$F^{-1}(\hat{f}(\omega)) = f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(\omega) e^{ix\omega} dx,$$

где $f(\omega)$ — ***спектральная плотность*** или просто ***спектр*** сигнала.

Спектр некоторого периодического сигнала представляет собой дискретное множество гармонических колебаний, в сумме дающее исходный

сигнал. Разложение некоторого сигнала на составляющие называется спектральным. График зависимости параметров сигнала от частоты называется спектральной диаграммой. Спектр сигнала - совокупность простых составляющих сигнала с определенными амплитудами, частотами и начальными фазами. Изменение формы сигнала приводит к изменению его спектра (и наоборот).

Теорема Котельникова (теорема Найквиста - Шеннона, или теорема отсчетов) гласит, что если аналоговый сигнал x(t) имеет конечный (ограниченный по ширине) спектр, то он может быть однозначно восстановлен без потерь по своим отсчетам, взятым с частотой, строго большей удвоенной верхней частоты $f_c : f > 2 \cdot f_c$.

Для преобразования цифровых данных в аналоговый сигнал используются следующие основные технологии модуляции (или кодирования):

- амплитудная (Amplitude Shift Keying, ASK);
- частотная (Frequencv Shift Keying, FSK);
- фазовая (Phase-Shifi Keving, PSK).

2.1. Амплитудная модуляция

В этом типе модуляции представлению нуля или единицы соответствует наличие или отсутствие несущей частоты при постоянной амплитуде. Результирующий сигнал при этом имеет вид

$$s(t) = \begin{cases} A\cos(2\pi f_c t), \text{ кодирует двоичную 1,} \\ 0, \text{кодирует двоичный 0,} \end{cases}$$

где $A\cos(2\pi f_c t)$ - несущий сигнал; A- амплитуда ; f_c - несущая частота;t- время.

2.2. Частотная модуляция

Частотная модуляция бывает:

• бинарной (Binary FSK, BFSK),

• многочастотной (Multiple FSK, MFSK).

В первом случае два двоичных числа модулируются сигналами двух различных частот, расположенных около несущей. Результирующий сигнал при этом имеет вид

$$s(t) = \begin{cases} A\cos(2\pi f_1 t) \\ A\cos(2\pi f_2 t) \end{cases}$$

Где f_1 и f_2 частоты, смещенные от несущей частоты f_c на величины равные по модулю, но противоположные по знаку.

Во втором случае, за один раз пересылается более одного бита за счет использования нескольких частот для модуляции сигнала. Сигнала при этом имеет вид

$$s_i = A\cos(2\pi f_i t), 1 \ll i \ll M,$$
$$f_i = f_c + (2i - 1 - M) f_d, M = 2^L$$

Где $A\cos(2\pi f_i t)$ сигнал; f_d - разностная частота; M — число различных сигнальных посылок , L- количество бит, переданных за один раз. Бинарная частотная модуляция менее восприимчива к ошибкам, чем амплитудная модуляция Многочастотная модуляция эффективнее бинарном. но к более подвержена ошибкам.

2.3. Фазовая модуляция

При фазовой модуляции для представления данных выполняется смещение несущего сигнала.

Фазовая модуляция бывает:

- двухуровневой (Binary PSK, BPSK);
- дифференциальной (Differential PSK, DPSK);
- квадратурной (Quadrature Phase-Shift Keying, QPSK);
- многоуровневой (Multiple FSK, MFSK).

В первом случае для представления двух двоичных цифр используются две фазы. Для одного периода передачи бита результирующий сигнал имеет вид:

$$s(t) = \begin{cases} A\cos(2\pi f_c t) \\ A\cos(2\pi f_c t + \pi) \end{cases} = \begin{cases} A\cos(2\pi f_c t), \text{ кодирует двоичную 1,} \\ -A\cos(2\pi f_c t), \text{кодирует двоичный 0.} \end{cases}$$

Во втором случае для представления двоичного нуля используется сигнал, фаза которого совпадает с фазой предыдущего сигнала, а для представления двоичной единицы - сигнал с фазой, противоположной фазе предыдущего. Такая схема называется дифференциальной, поскольку сдвиг фаз выполняется относительно предыдущего переданного бита, а не относительно какого-то эталонного сигнала.

В третьем случае за раз передается более одного бита, при этом вместо сдвига фазы на к, как в двухуровневой модуляции, используются сдвиги фаз, кратные $\pi/2$:

$$s(t) = \begin{cases} A\cos(2\pi f_c t + \frac{\pi}{4}) \text{ кодирует двоичную 11,} \\ A\cos(2\pi f_c t + \frac{3\pi}{4}) \text{ кодирует двоичную 10,} \\ A\cos(2\pi f_c t + \frac{5\pi}{4}) \text{ кодирует двоичную 00,} \\ A\cos(2\pi f_c t + \frac{7\pi}{4}) \text{ кодирует двоичную 01.} \end{cases}$$

Схема работы многоуровневой фазовой модуляции схожа со схемой работы квадратурной фазовой модуляции, но в каждый момент времени передается по три бита, используется восемь различных углов сдвига фаз, для каждого угла используется несколько амплитуд.

2.4. Квадратурная амплитудная модуляция

Схема работы квадратурной амплитудной модуляции (QAM) использует принципы функционирования амплитудной и фазовой модуляций. Два различных сигнала передаются одновременно на одной несущей частоте, но при этом задействованы две се амплитудно-модулированные копии, сдвинутые относительно друг друга на 90^0 (т.е. находящиеся в квадратуре). Амплитуды копий несущей меняются дискретно, что приводит к образованию сигнала с дискретным изменением одновременно и амплитуды, и фазы. Приемник полученные сигналы демодулирует и объединяет с целью восстановления исходного двоичного сигнала.

Исходя из таких соображений, фазовую модуляцию можно рассматривать, как частный случай квадратурной амплитудной модуляции.

В случае двухуровневой амплитудной модуляции (2QAM) каждый из двух потоков может находиться в одном из двух состояний, а объединенный поток в одном из четырех. В случае четырехуровневой модуляции (те четырех различных уровней амплитуды. 4QAM) объединенный поток будет находиться уже в одном из 16 состояний. Чем больше число состояний. тем выше скорость передачи данных, возможная при определенной ширине полосы пропускания. Но чем больше число состоянии, тем выше потенциальная частота возникновения ошибок из-за помех или поглощения.

2.5 Технология расширенного спектра

Основная идея метода состоит в том, чтобы распределить информационный сигнал по широкой полосе радиодиапазона, что в итоге позволит значительно усложнить подавление или перехватить сигнал.

Расширение спектра скачкообразной перестройкой частоты Frequency

Hopping Spread Spectrum, FHSS). Передача ведется с постоянном сменой несущей в пределах широкого диапазона частот. В результате мощность сигнала распределяется по всему диапазону, а прослушивание какой-то определенной частоты дает только небольшой шум. Последовательность несущих частот псевдослучайна и известна только передатчику и приемнику. Попытка подавления сигнала в каком-то узком диапазоне почти не ухудшает сигнал, так как подавляется только небольшая часть информации.

На каждой несущей частоте для передачи дискретной информации применяются стандартные методы модуляции — частотная или фазовая. Для синхронизации приемника и передатчика в течение некоторого времени передаются синхронизирующие последовательности бит. Несущая частота меняется в соответствии с номерами частотных подканалов, вырабатываемых алгоритмом псевдослучайных чисел. Если частота смены подканалов ниже, чем скорость передачи данных в канале, то такой режим называют медленным расширением спектра, в противном случае — быстрым расширением спектра. Метод быстрого расширения спектра более устойчив к помехам, так как помехи, подавляющие сигнал в определенном подканале, не приводят к потере бита, поскольку его значение повторяется несколько раз в различных частотных подканалах. Метод медленного расширения спектра менее устойчив к помехам, но его проще реализовать.

Прямое последовательное расширение спектра (Direct Sequence Spread Spectrum, DSSS). В методе прямого последовательного расширения спектра, в отличие от метода расширения спектра скачкообразной перестройкой частоты, весь частотный диапазон занимается не за счет постоянных переключений с частоты на частоту, а за счет того, что каждый бит информации заменяется последовательностью из N бит, что дает увеличение тактовой скорости передачи сигналов в N раз и соответствующее расширение в N раз спектра сигнала.

Передача двоичной единицы заменяется передачей расширяющей последовательности. Двоичный нуль кодируется инверсным значением расширяющей последовательности. Количество бит в расширяющей последовательности определяет коэффициент расширения исходного кода. Для кодирования битов результирующего кода может использоваться любой вид модуляции. Чем больше коэффициент расширения, тем шире спектр результирующего сигнала и выше степень подавления помех. Но при этом растет занимаемый каналом диапазон спектра.

Помехи искажают только определенные частоты спектра сигнала, поэтому приемник с большой степенью вероятности может правильно распознавать передаваемую информацию.

Метод прямого последовательного расширения спектра в меньшей степени защищен от помех, чем метод быстрого расширения спектра, так как мощные помехи влияют на часть спектра, а значит, и на результат распознавания единиц или нулей.

2.6. Кодирование сигнала

Одной из основных задач физического уровня модели OSI является преобразование данных в электромагнитные сигналы, и наоборот. Переход от электромагнитных импульсов к последовательности бит называют кодированием сигнала.

Рассмотрим наиболее распространенные методы кодирования.

2.6.1. Код NRZ и NRZI

Код NRZ (Non Return to Zero – без возврата к нулю) – это простейший код, представляющий собой обычный цифровой сигнал. Логическому нулю соответствует высокий уровень напряжения в кабеле, логической единице – низкий уровень напряжения (или наоборот, что не принципиально). Уровни могут быть разной полярности (положительной и отрицательной) или же одной полярности (положительной или отрицательной) (*Рис. Л.2а*). В течение битового интервала (bit time, BT), то есть времени передачи одного бита никаких изменений уровня сигнала в кабеле не происходит.

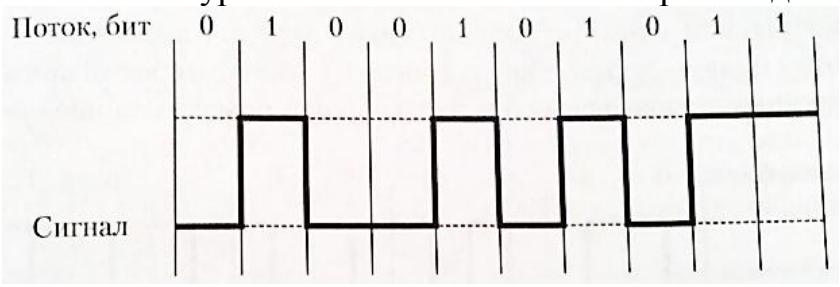

Рис. Л.2а.- Код NRZ

К несомненным достоинствам кода NRZ относятся его довольно простая реализация (исходный сигнал не надо ни специально кодировать на передающем конце, ни декодировать на приемном конце), а также минимальная среди других кодов пропускная способность линии связи, требуемая при данной скорости передачи. Ведь наиболее частое изменение сигнала в сети будет при непрерывном чередовании единиц и нулей, то есть при последовательности 1010101010..., поэтому при скорости передачи, равной 10 Мбит/с (длительность одного бита равна 100 нс) частота изменения сигнала и соответственно требуемая пропускная способность линии составит 1 / 200нс = 5 МГц (рис *Рис. Л.2б*).

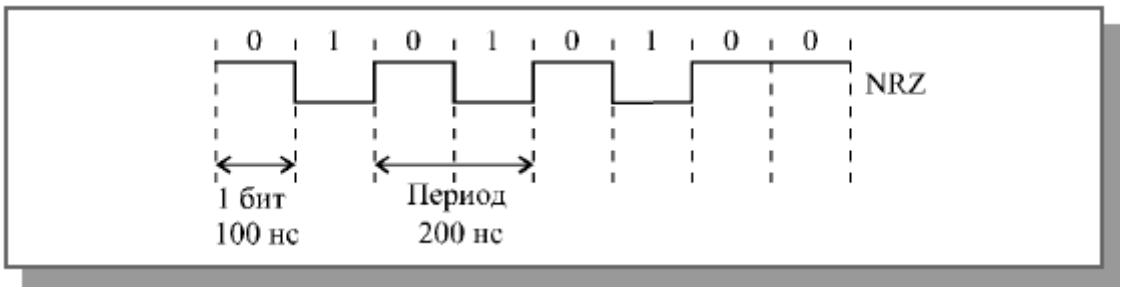

Рис. Л.2б.- Скорость передачи и требуемая пропускная способность при коде NRZ

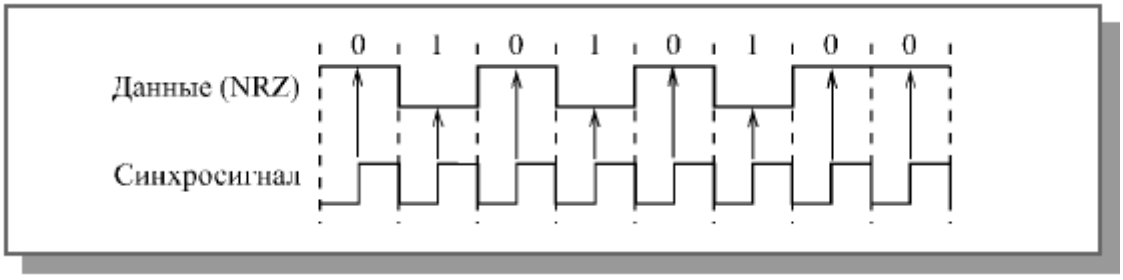

Рис. Л.2в.- Передача в коде NRZ с синхросигналом

Самый большой недостаток кода NRZ – это возможность потери синхронизации приемником во время приема слишком длинных блоков (пакетов) информации. Приемник может привязывать момент начала приема только к первому (стартовому) биту пакета, а в течение приема пакета он вынужден пользоваться только внутренним тактовым генератором (внутренними часами). Например, если передается последовательность нулей или последовательность единиц, то приемник может определить, где проходят границы битовых интервалов, только по внутренним часам. И если часы приемника расходятся с часами передатчика, то временной сдвиг к концу приема пакета может превысить длительность одного или даже нескольких бит. В результате произойдет потеря переданных данных. Так, при длине пакета в 10000 бит допустимое расхождение часов составит не более 0,01% даже при идеальной передаче формы сигнала по кабелю.

Во избежание потери синхронизации, можно было бы ввести вторую линию связи для синхросигнала (рис. *Л.2в*). Но при этом требуемое количество кабеля, число приемников и передатчиков увеличивается в два раза. При большой длине сети и значительном количестве абонентов это невыгодно.

В связи с этим код NRZ используется только для передачи короткими пакетами (обычно до 1 Кбита).

Большой недостаток кода NRZ состоит еще и в том, что он может обеспечить обмен сообщениями (последовательностями, пакетами) только фиксированной, заранее обговоренной длины. Дело в том, что по принимаемой информации приемник не может определить, идет ли еще передача или уже закончилась. Для синхронизации начала приема пакета используется стартовый служебный бит, чей уровень отличается от пассивного состояния линии связи (например, пассивное состояние линии при отсутствии передачи – 0, стартовый бит – 1). Заканчивается прием после отсчета приемником заданного количества бит последовательности (рис. *Л.2г*).

Рис. Л.2г.- Определение окончания последовательности при коде NRZ

Наиболее известное применение кода NRZ – это стандарт RS232-C, последовательный порт персонального компьютера. Передача информации в нем ведется байтами (8 бит), сопровождаемыми стартовым и стоповым битами.

Три остальных кода (RZ, манчестерский код, бифазный код) принципиально отличаются от NRZ тем, что сигнал имеет дополнительные переходы (фронты) в пределах битового интервала. Это сделано для того, чтобы приемник мог подстраивать свои часы под принимаемый сигнал на каждом битовом интервале. Отслеживая фронты сигналов, приемник может точно синхронизовать прием каждого бита. В результате небольшие расхождения часов приемника и передатчика уже не имеют значения. Приемник может надежно принимать последовательности любой длины. Такие коды называются самосинхронизирующимися. Можно считать, что самосинхронизирующиеся коды несут в себе синхросигнал.

Код NRZI (Non Return to Zero Invert to ones) представляет собой модификацию кода NRZ. В этом двухуровневом коде принимается во внимание значение предыдущего бита. Уровень сигнала меняется, если текущий бит - единица, и повторяет предыдущий, если текущий бит имеет значение 0 (рис. *Л.3*). NRZI используется в основном для работы с оптоволоконной средой, в сетях 100BASE-FX.

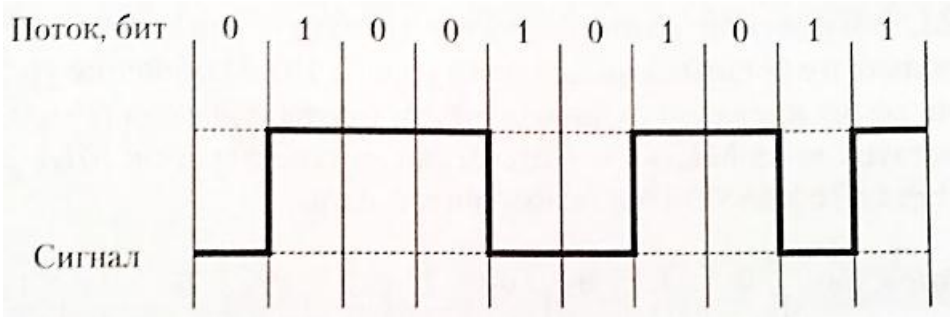

Рис. Л.3.- Код NRZI

2.6.2. Код Rz

Код RZ (Return to Zero – с возвратом к нулю) – этот трехуровневый код получил такое название потому, что после значащего уровня сигнала в первой половине битового интервала следует возврат к некоему "нулевому", среднему уровню (например, к нулевому потенциалу). Переход к нему происходит в середине каждого битового интервала. Логическому нулю, таким образом, соответствует положительный импульс, логической единице – отрицательный (или наоборот) в первой половине битового интервала (*Рис. Л.4а*). Код RZ нашел применение в оптоволоконных сетях .

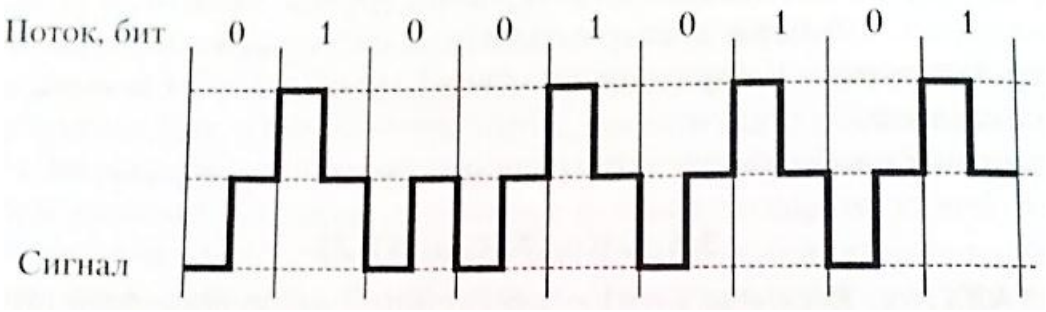

Рис. Л.4а.- Код Rz

В центре битового интервала всегда есть переход сигнала (положительный или отрицательный), следовательно, из этого кода приемник легко может выделить синхроимпульс (строб). Возможна временная привязка не только к началу пакета, как в случае кода NRZ, но и к каждому отдельному биту, поэтому потери синхронизации не произойдет при любой длине пакета.

Еще одно важное достоинство кода RZ – простая временная привязка приема, как к началу последовательности, так и к ее концу. Приемник просто должен анализировать, есть изменение уровня сигнала в течение битового интервала или нет (*Рис. Л.4б*).

Первый битовый интервал без изменения уровня сигнала соответствует окончанию принимаемой последовательности бит. Поэтому в коде RZ можно использовать передачу последовательностями переменной длины.

Рис. Л.4б.- Определение начала и конца приема при коде RZ

Недостаток кода RZ состоит в том, что для него требуется вдвое большая полоса пропускания канала при той же скорости передачи по сравнению с NRZ (так как здесь на один битовый интервал приходится два изменения уровня сигнала). Например, для скорости передачи информации 10 Мбит/с требуется пропускная способность линии связи 10 МГц, а не 5 МГц, как при коде NRZ (*Рис. Л.4в*).

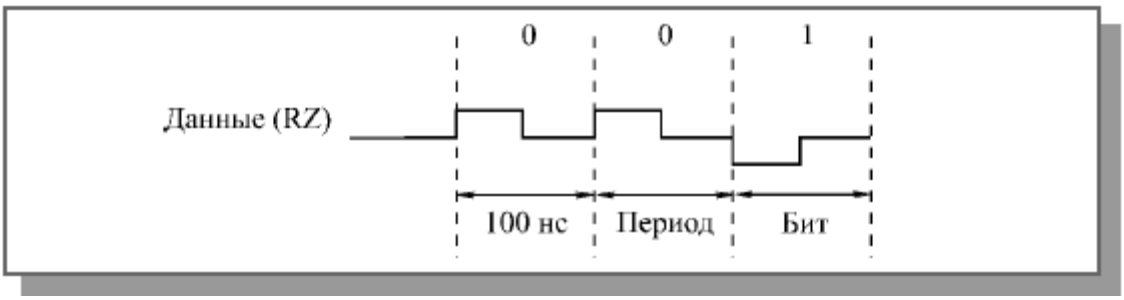

Рис. Л.4в - Скорость передачи и пропускная способность при коде RZ

Другой важный недостаток – наличие трех уровней, что всегда усложняет аппаратуру как передатчика, так и приемника.

Код RZ применяется не только в сетях на основе электрического кабеля, но и в оптоволоконных сетях. Правда, в них не существует положительных и отрицательных уровней сигнала, поэтому используется три следующие уровня: отсутствие света, "средний" свет, "сильный" свет. Это очень удобно: даже когда нет передачи информации, свет все равно

присутствует, что позволяет легко определить целостность оптоволоконной линии связи без дополнительных мер (*Рис. Л.4г*).

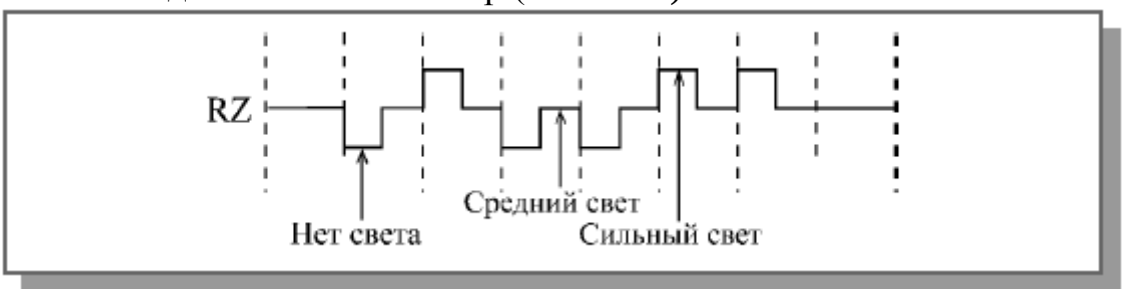

Рис. Л.4г - Использование кода RZ в оптоволоконных сетях

2.6.3. Манчестерский код

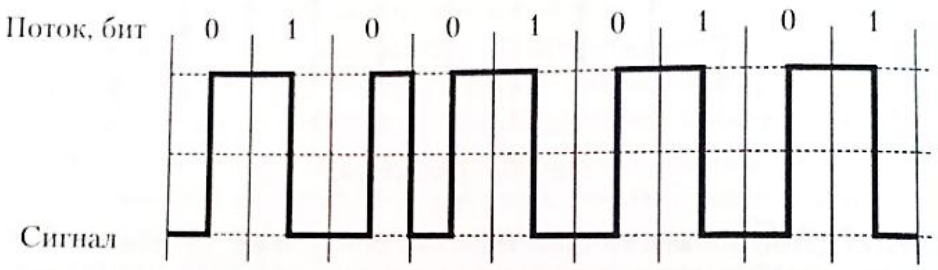

Рис. Л.5а.- Манчестерский код

Манчестерский код (или код Манчестер-II) получил наибольшее распространение в локальных сетях. Он также относится к самосинхронизирующимся кодам, но в отличие от RZ имеет не три, а всего два уровня, что способствует его лучшей помехозащищенности и упрощению приемных и передающих узлов. Логическому нулю соответствует положительный переход в центре битового интервала (то есть первая половина битового интервала – низкий уровень, вторая половина – высокий), а логической единице соответствует отрицательный переход в центре битового интервала (или наоборот) (рис. *Л.5а*.).

Как и в RZ, обязательное наличие перехода в центре бита позволяет приемнику манчестерского кода легко выделить из пришедшего сигнала синхросигнал и передать информацию сколь угодно большими последовательностями без потерь из-за рассинхронизации. Допустимое расхождение часов приемника и передатчика может достигать 25%.

Подобно коду RZ, при использовании манчестерского кода требуется пропускная способность линии в два раза выше, чем при применении простейшего кода NRZ. Например, для скорости передачи 10 Мбит/с требуется полоса пропускания 10 МГц (рис. Л5б).

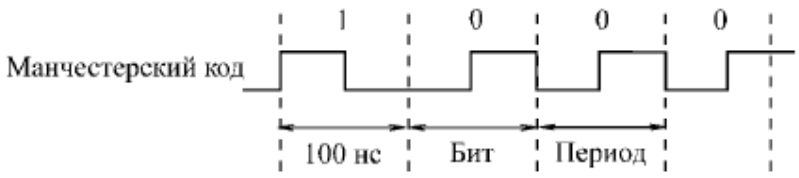

Рис. Л5б.- Скорость передачи и пропускная способность при манчестерском коде

Как и при коде RZ, в данном случае приемник легко может определить не только начало передаваемой последовательности бит, но и ее конец. Если в течение битового интервала нет перехода сигнала, то прием заканчивается. В манчестерском коде можно передавать последовательности бит переменной длины (*рис. Л.5в*). Процесс определения времени передачи называют еще контролем несущей, хотя в явном виде несущей частоты в данном случае не присутствует.

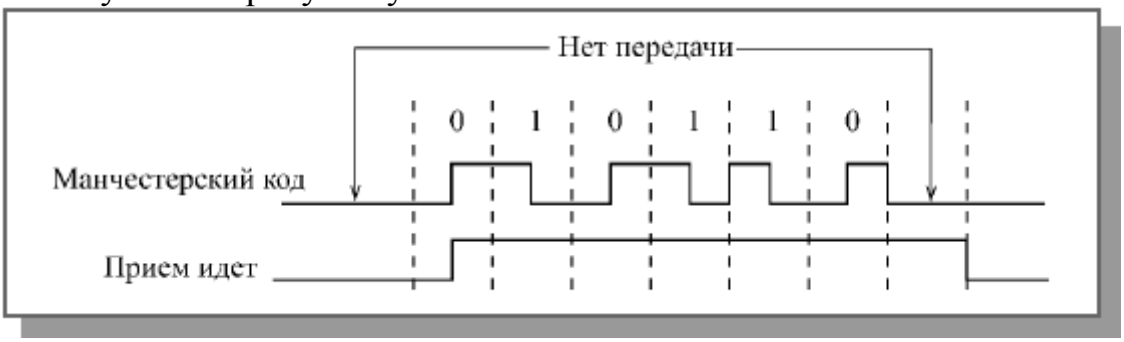

Рис. Л5в.- Определение начала и конца приема при манчестерском коде

Манчестерский код используется как в электрических, так и в оптоволоконных кабелях (в последнем случае один уровень соответствует отсутствию света, а другой – его наличию).

Основное достоинство манчестерского кода – постоянная составляющая в сигнале (половину времени сигнал имеет высокий уровень, другую половину – низкий). Постоянная составляющая равна среднему значению между двумя уровнями сигнала.

Если высокий уровень имеет положительную величину, а низкий – такую же отрицательную, то постоянная составляющая равна нулю. Это дает возможность легко применять для гальванической развязки импульсные трансформаторы. При этом не требуется дополнительного источника питания для линии связи (как, например, в случае использования оптронной гальванической развязки), резко уменьшается влияние низкочастотных помех, которые не проходят через трансформатор, легко решается проблема согласования.

Если же один из уровней сигнала в манчестерском коде нулевой (как, например, в сети Ethernet), то величина постоянной составляющей в течение передачи будет равна примерно половине амплитуды сигнала. Это позволяет легко фиксировать столкновения пакетов в сети (конфликт, коллизию) по отклонению величины постоянной составляющей за установленные пределы.

Частотный спектр сигнала при манчестерском кодировании включает в себя только две частоты: при скорости передачи 10 Мбит/с это 10 МГц (соответствует передаваемой цепочке из одних нулей или из одних единиц) и 5 МГц (соответствует последовательности из чередующихся нулей и единиц: 1010101010...). Поэтому с помощью простейших полосовых фильтров можно легко избавиться от всех других частот (помехи, наводки, шумы).

2.6.4. Код MLT-3

Код MLT-3 (Multi Level Transmission-3) - трехуровневый код. Как и в NRZI, логической единице соответствует смена уровня сигнала а при передаче нуля сигнал не меняется (рис. 3.10). Изменение уровня сигнала происходит последовательно с учетом предыдущего перехода Основной недостаток кода MLT-3 - отсутствие синхронизации. MLT-3 применяется в сетях 100BASE-T на основе витой пары.

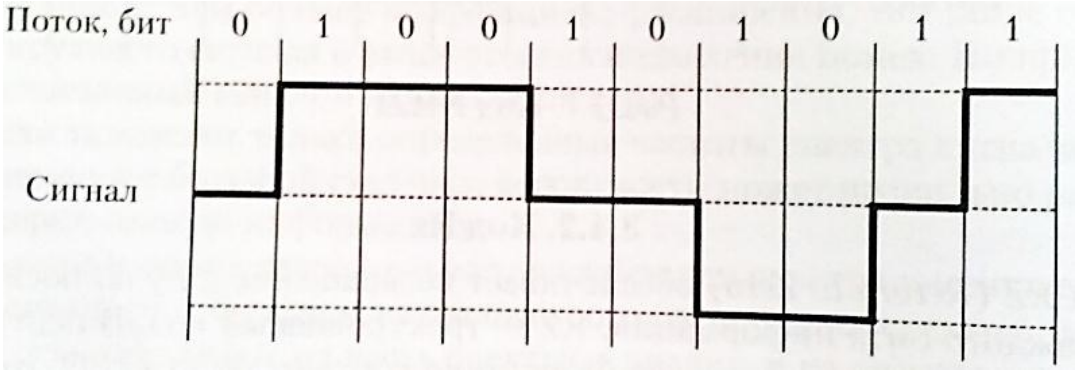

Рис. Л.6.- **Код MLT-3**

Порядок выполнения работы

1. Ознакомиться с теоретическим материалом п.2 «Модуляция сигналов».
2. Выполнить задания пунктов лабораторной работы 1. Получить соответствующие графики.
3. Согласно таблице 1 в приложении 1 перевести свою фамилию в восьмеричный код, а затем уже в двоичный. Получившуюся последовательность бит закодировать методами NRZ, RZ и манчестерским кодом.
4. Составить отчёт о выполненной работе, включив в него листинги программ и полученные графики.

Контрольные вопросы.

1. Способ кодирования информации методом NRZ.
2. Достоинства и недостатки кода NRZ.
3. Способ кодирования информации методом RZ.
4. Достоинства и недостатки кода RZ.
5. Способ кодирования информации манчестерским кодом.
6. Достоинства и недостатки кода манчестерского кода.

Разложение импульсного сигнала в частичный ряд Фурье. В цифровой технике основным типом сигналов является импульсный сигнал.

Импульсный сигнал можно описать математически в синусоидальной форме. Такой тип сигнала называется меандром.

Меандр — бесконечный, периодический сигнал прямоугольной формы (импульсный сигнал), широко используемый в радиотехнике. Длительность импульса и длительность паузы в периоде такого сигнала равны.

Спектр меандра имеет вид:

$$s(t)=\frac{A}{2}+\frac{2A}{\pi}\left(\cos\left(\frac{2\pi}{T}t\right)-\frac{1}{3}\cos\left(3\frac{2\pi}{T}t\right)+\frac{1}{5}\cos\left(5\frac{2\pi}{T}t\right)-\ldots\right).$$

Гармоники, образующие меандр, имеют амплитуду, обратно пропорциональную номеру соответствующей гармоники.

Задание для выполнения

Разработать код *m*-файла, результатом выполнения которого являются графики меандра (рис. Л.7), реализованные с различным количеством гармоник.

Листинг программы в Matlab:

```
% meandr.m
clear all
close all
clc
% количество отсчетов:
N=8;
% частота дискретизации:
t=-1:0.01:1;
% значение амплитуды:
A=1;
T=1;
nh=(1:N)*2-1;
% входной сигнал:
harmonics=cos(2*pi*nh'*t/T);
Am=2/pi./nh;
Am(2:2:end)=-Am(2:2:end);
s1=harmonics.*repmat(Am',1,length(t));
s2=cumsum(s1);
for k=1:N
subplot(4,2,k)
plot(t, s2(k,:)),grid
end
```

Здесь функция *repmat(A,M,N)* формирует массив из частей; имеет три входных аргумента: массив *A*, количество строк *M* и столбцов *N* для вновь создаваемого массива; *cumsum* — суммирование элементов массива с накоплением.

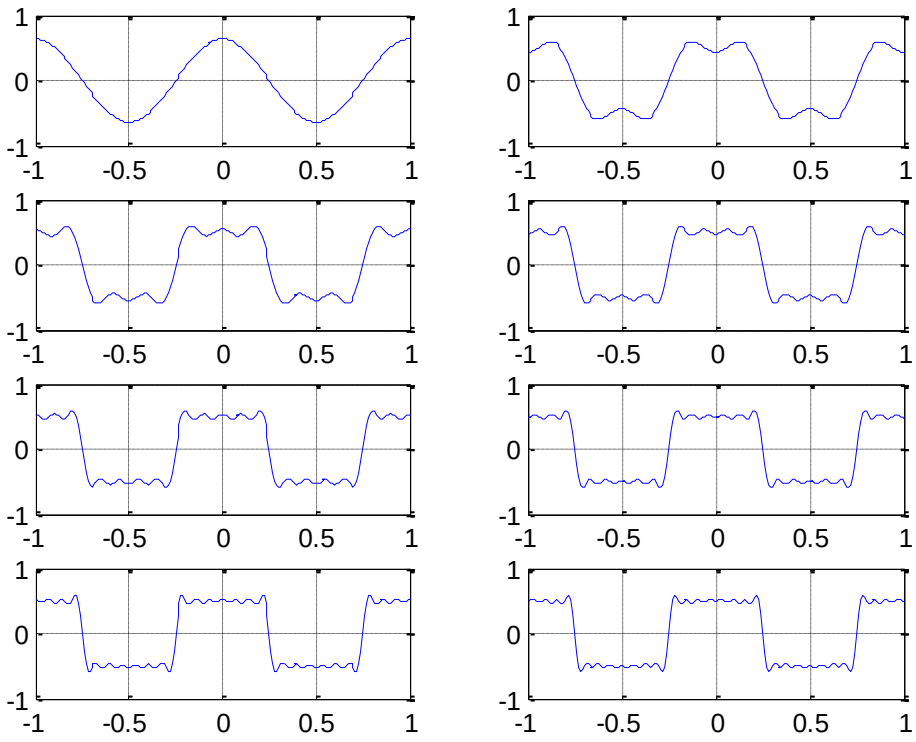

Рис.Л.7. **Графики меандра, содержащего различное число гармоник:**

на всех графиках вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – время, деленное на период сигнала

Определение спектра и параметров сигнала

Задание. Определить спектр двух отдельных сигналов и их суммы. Частота дискретизации (количество отсчётов) выбирается на основе теоремы Котельникова как удвоенная ширина спектра исходного сигнала (таким образом, в следующем примере достаточно было взять частоту дискретизации 80 Гц).

Попробуйте выполнить задание с другой частотой дискретизации. Что будет, если взять частоту дискретизации меньше 80 Гц?

Для двух синусоидальных сигналов (рис. Л.8) требуется определить их спектр. В файле *spectre.m* задаем параметры сигналов:

```
% spectre.m
clear all
close all
clc
mkdir 'signal';
mkdir 'spectre';
tmax = 0.5;% Длина сигнала (с)
fd = 512; % Частота дискретизации (Гц) (количество отсчётов)
f1 = 10;% Частота первого сигнала (Гц)
```

```
f2 = 40;% Частота второго сигнала (Гц)
a1 = 1;% Амплитуда первого сигнала
a2 = 0.7;% Амплитуда второго сигнала
fd2 = fd/2; % Спектр сигнала
% Рассмотрим два сигнала (синусоиды) разной частоты
t = 0:1./fd:tmax; % Массив отсчётов времени
signal1 = a1*sin(2*pi*t*f1);
signal2 = a2*sin(2*pi*t*f2);
figure()
plot(signal1,'b'); % голубая
hold on
plot(signal2,'r'); % красная
hold off
title('Signal');
grid on
```

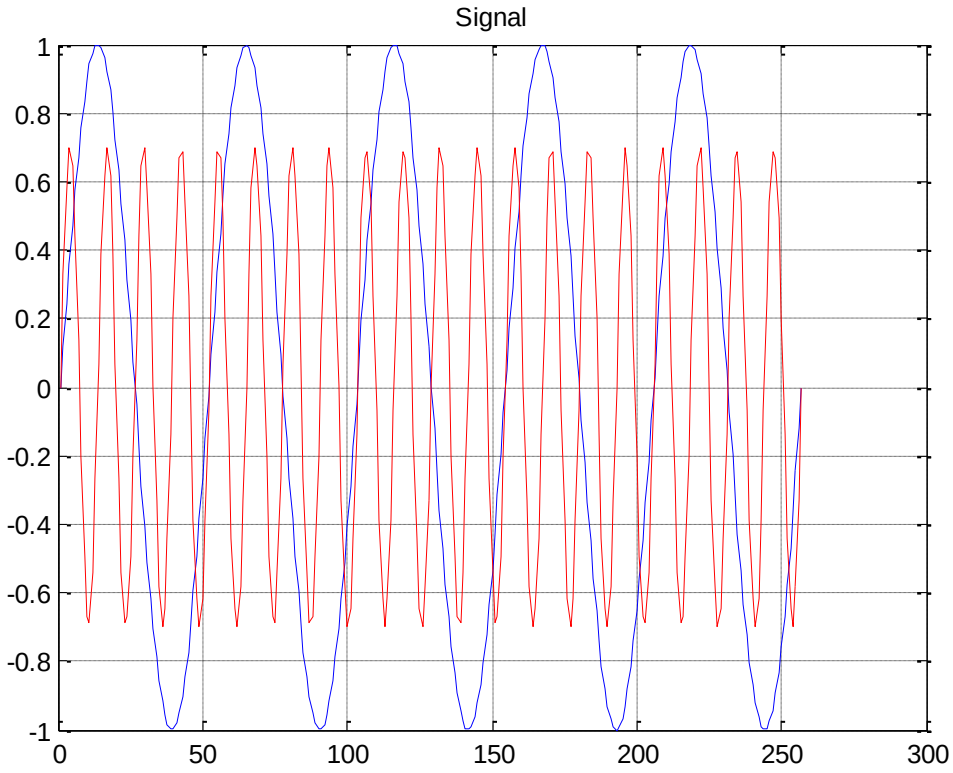

Рис. Л.8. **Два синусоидальных сигнала разной частоты:** вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – частота, измеряемая в герцах (Гц)

С помощью быстрого преобразования Фурье найдем спектры сигналов (рис. Л.9), добавив в файл *spectre.m* следующий код.

```
% Посчитаем спектр
% Амплитуды преобразования Фурье сигнала 1
spectre1 = abs(fft(signal1,fd));
% Амплитуды преобразования Фурье сигнала 2
spectre2 = abs(fft(signal2,fd));
```

```
% Построение спектров сигналов
figure()
plot(spectre1,'b'); % голубая
hold on
plot(spectre2,'r'); % красная
hold off
title('Spectre');
```

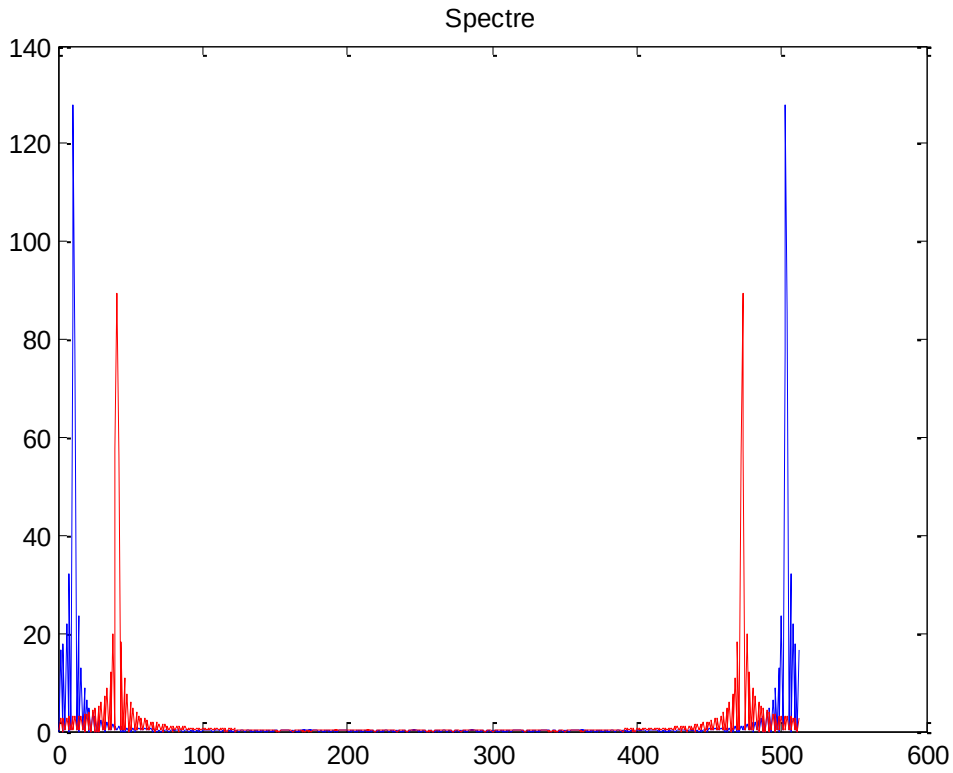

Рис. Л.9. **График спектров синусоидальных сигналов**
вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – частота, измеряемая в герцах (Гц)

Учитывая реализацию преобразования Фурье, скорректируем график спектра (рис. Л.10): отбрасываются дублирующие отрицательные частоты, а также учитывается то, что на каждом шаге вычисления быстрого преобразования Фурье происходит суммирование амплитуд сигналов.

Добавляем в файл *spectre.m* следующий код.

```
% Исправление графика спектра
f = 1000*(0:fd2)./(2*fd); % Сетка частот
% Нормировка спектров по амплитуде
spectre1 = 2*spectre1/fd2;
spectre2 = 2*spectre2/fd2;
% Построение спектров сигналов
figure()
plot(f,spectre1(1:fd2+1),'b'); % голубая
```

```
hold on
plot(f,spectre2(1:fd2+1),'r'); % красная
hold off
xlim([0 100]);
title('Fixed spectre');
xlabel('Frequency (Hz)');
```

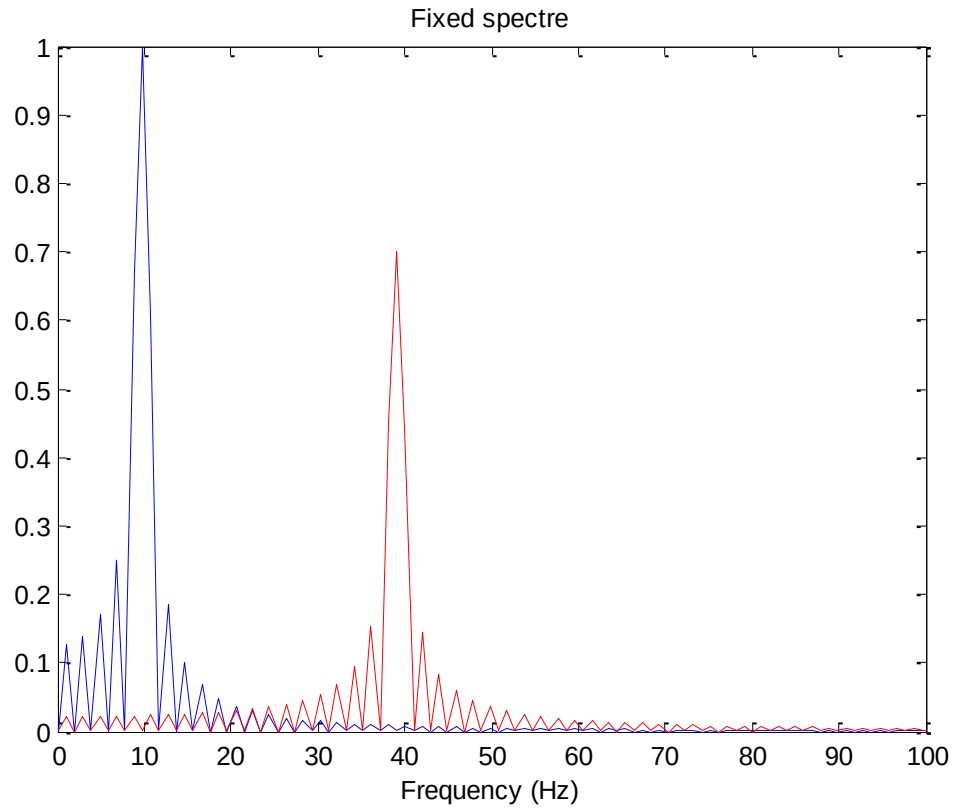

Рис. Л.10. **Исправленный график спектров синусоидальных сигналов**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – частота, измеряемая в герцах (Гц)

Аналогично найдем спектр суммы рассмотренных сигналов (рис Л.11), создав файл *spectre_sum.m* со следующим кодом.

```
% spectre_sum.m
clear all
close all
clc
mkdir 'signal';
mkdir 'spectre';
% Длина сигнала (с)
tmax = 0.5;
% Частота дискретизации (Гц) (количество отсчётов)
fd = 512;
% Частота первого сигнала (Гц)
f1 = 10;
% Частота второго сигнала (Гц)
f2 = 40;
```

```
% Амплитуда первого сигнала
a1 = 1;
% Амплитуда второго сигнала
a2 = 0.7;
% Спектр сигнала
fd2 = fd/2;
% Сумма двух сигналов (синусоиды) разной частоты
% Массив отсчётов времени:
t = 0:1./fd:tmax;
signal1 = a1*sin(2*pi*t*f1);
signal2 = a2*sin(2*pi*t*f2);
signal = signal1 + signal2;
figure()
plot(signal), grid;
title('Signal');

% Подсчет спектра:
% Амплитуды преобразования Фурье сигнала
spectre = fft(signal,fd);
% Сетка частот
f = 1000*(0:fd2)./(2*fd);
% Нормировка спектра по амплитуде:
spectre = 2*sqrt(spectre.*conj(spectre))./fd2;
% Построение спектра сигнала
figure()
plot(f,spectre(1:fd2+1)), grid
xlim([0 100]);
title('Spectre');
xlabel('Frequency (Hz)');
```

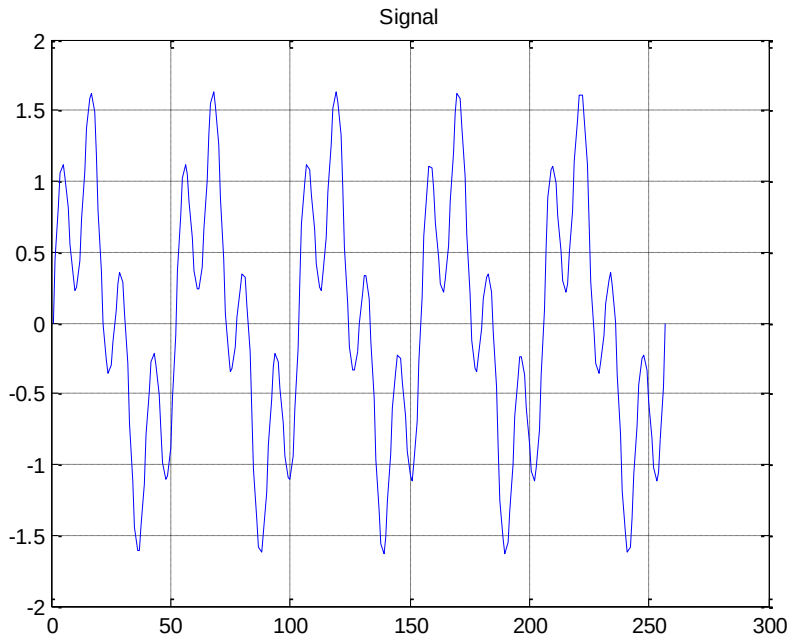

Рис. Л.11. **Суммарный сигнал**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – частота, измеряемая в герцах (Гц)

В результате получим аналогичный предыдущему результат (рис. Л.12), т.е. спектр суммы сигналов равен сумме спектров сигналов, что вытекает из свойств преобразования Фурье.

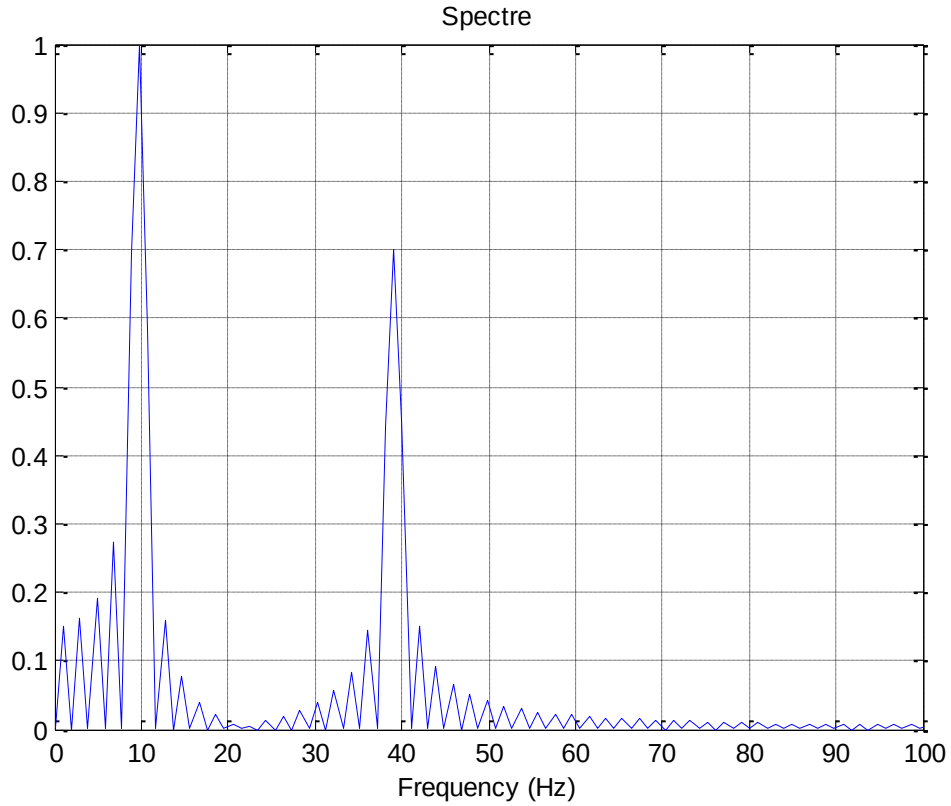

Рис. Л.12. **Спектр суммарного сигнала**
вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах;
горизонтальная ось – частота, измеряемая в герцах (Гц)

Демонстрация принципов модуляции сигнала на примере аналоговой амплитудной модуляции.

Следующий код в файле *am.m* демонстрирует принципы модуляции сигнала на примере аналоговой амплитудной модуляции (рис. Л.13).

```
% am.m
clear all
close all
clc
mkdir 'signal';
mkdir 'spectre';
% Модуляция синусоид с частотами 50 и 5
% Длина сигнала (с)
tmax = 0.5;
% Частота дискретизации (Гц) (количество отсчётов)
fd = 512;
% Частота сигнала (Гц)
f1 = 5;
% Частота несущей (Гц)
```

```
f2 = 50;
% Спектр сигнала
fd2 = fd/2;
% Построение графиков двух сигналов (синусоиды)
% разной частоты
% Массив отсчётов времени:
t = 0:1./fd:tmax;
signal1 = sin(2*pi*t*f1);
signal2 = sin(2*pi*t*f2);
signal = signal1 .* signal2;
figure()
plot(signal, 'b'), grid;
hold on
% Построение огибающей:
plot(signal1, 'r'), grid;
plot(-signal1, 'r'), grid;
hold off
title('Signal');

% Расчет спектра:
% Амплитуды преобразования Фурье-сигнала
spectre = fft(signal,fd);
% Сетка частот
f = 1000*(0:fd2)./(2*fd);
% Нормировка спектра по амплитуде:
spectre = 2*sqrt(spectre.*conj(spectre))./fd2;
% Построение спектра:
figure()
plot(f,spectre(1:fd2+1), 'b'), grid;
xlim([0 100]);
title('Spectre');
xlabel('Frequency (Hz)');
```

В результате получаем, что спектр произведения есть свёртка спектров (рис. Л.14).

Кодирование сигнала. Исследование свойства самосинхронизации сигнала. По заданной выходной битовой последовательности требуется получить кодированный сигнал для нескольких кодов, проверить свойства самосинхронизуемости кода.

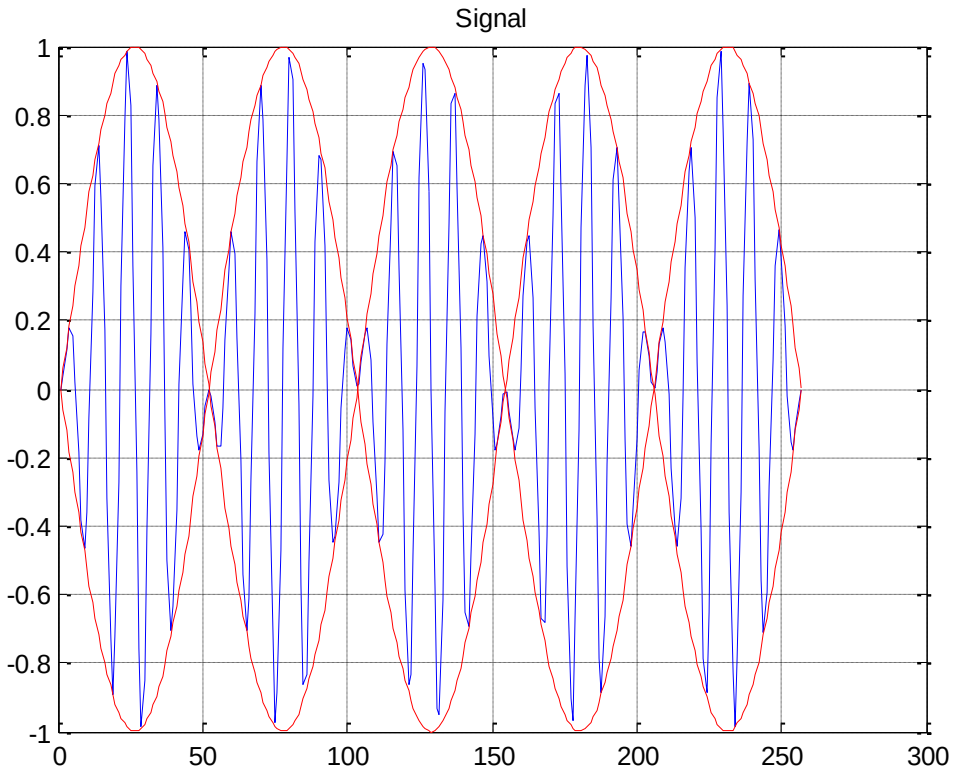

Рис. Л.13. **Сигнал и огибающая при амплитудной модуляции**
вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – частота, измеряемая в герцах (Гц)

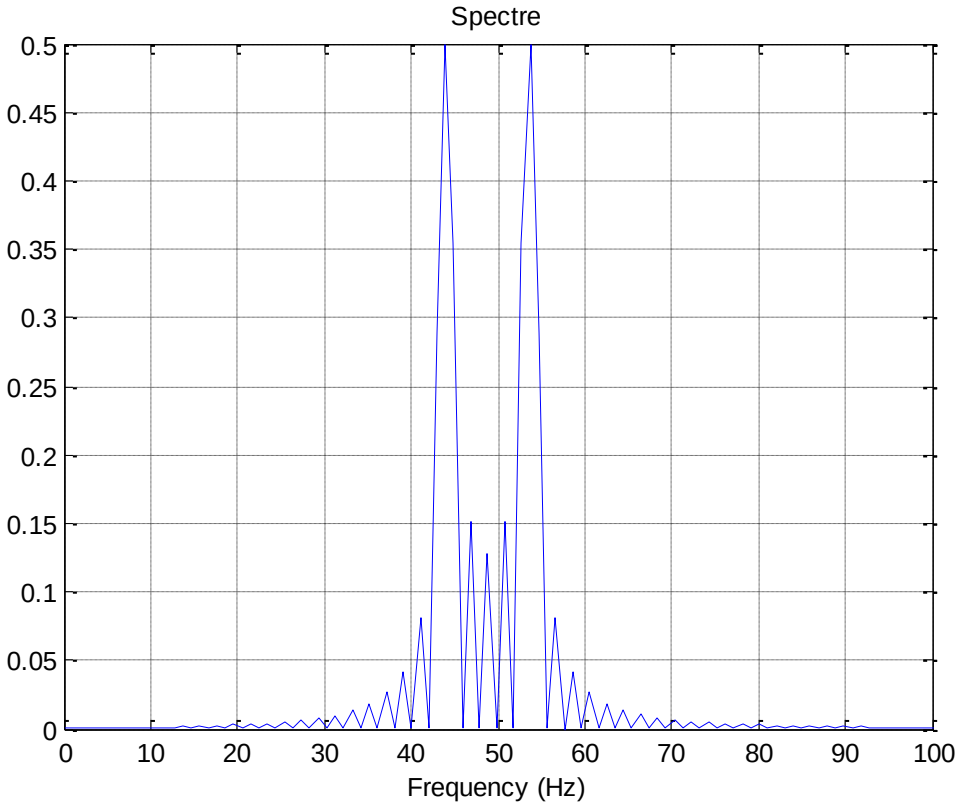

Рис. Л.14. **Спектр сигнала при амплитудной модуляции**
вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – частота, измеряемая в герцах (Гц)

Создаём файл *main.m*:

```
% main.m
clear all
close all
clc
% Задаем входную кодовую последовательность:
data=[0 1 0 0 1 1 0 0 0 1 1 0];
% Задаем входную кодовую последовательность
% для проверки свойства самосинхронизации:
data_sync=[0 0 0 0 0 0 0 1 1 1 1 1 1 1];
% Построение графиков кодированного сигнала
mkdir 'signal';
axis('auto');
% Униполярное кодирование
wave=unipolar(data);
figure()
plot(wave), grid;
ylim([-1 6]);
title('Unipolar');

% Кодирование ami
wave=ami(data);
figure()
plot(wave), grid;
ylim([-6 6]);
title('AMI');

% Кодирование NRZ
wave=bipolarnrz(data);
figure()
plot(wave), grid;
ylim([-6 6]);
title('Bipolar Non-Return to Zero');

% Кодирование RZ
wave=bipolarrz(data);
figure()
plot(wave), grid;
ylim([-6 6]);
title('Bipolar Return to Zero');

% Манчестерское кодирование
wave=manchester(data);
figure()
plot(wave), grid;
ylim([-6 6]);
title('Manchester');

% Дифференциальное манчестерское кодирование
wave=diffmanc(data);
figure()
plot(wave), grid;
ylim([-6 6]);
```

```
title('Differential Manchester');

% Построение графиков кодированного сигнала
% для проверки свойства самосинхронизации
mkdir 'sync';
axis('auto');
% Униполярное кодирование
wave=unipolar(data_sync);
figure()
plot(wave), grid;
ylim([-1 6]);
title('Unipolar');

% Кодирование AMI
wave=ami(data_sync);
figure()
plot(wave), grid;
ylim([-6 6]);
title('AMI');

% Кодирование NRZ
wave=bipolarnrz(data_sync);
figure()
plot(wave), grid;
ylim([-6 6]);
title('Bipolar Non-Return to Zero');

% Кодирование RZ
wave=bipolarrz(data_sync);
figure()
plot(wave), grid;
ylim([-6 6]);
title('Bipolar Return to Zero');

% Манчестерское кодирование
wave=manchester(data_sync);
figure()
plot(wave), grid;
ylim([-6 6]);
title('Manchester');

% Дифференциальное манчестерское кодирование
wave=diffmanc(data_sync);
figure()
plot(wave), grid;
ylim([-6 6]);
title('Differential Manchester');
```

Следующая функция (в отдельном файле *maptowave.m*) по входному битовому потоку строит график сигнала:

```
% maptowave.m
function wave=maptowave(data)
```

```
data=upsample(data,100);
wave=filter(5*ones(1,100),1,data);
```

Каждая функция преобразования кодовой последовательности находится в отдельном файле. Например, униполярное кодирование реализуется с помощью следующей функции:

```
% unipolar.m
function wave=unipolar(data)
wave=maptowave(data);
```

Кодирование AMI реализуется с помощью следующей функции:

```
% ami.m
function wave=ami(data)
am=mod(1:length(data(data==1)),2);
am(am==0)=-1;
data(data==1)=am;
wave=maptowave(data);
```

Кодирование NRZ реализуется с помощью следующей функции:

```
% bipolarnrz.m
function wave=bipolarnrz(data)
data(data==0)=-1;
wave=maptowave(data);
```

Кодирование RZ реализуется с помощью следующей функции:

```
% bipolarrz.m
function wave=bipolarrz(data)
data(data==0)=-1;
data=upsample(data,2);
wave=maptowave(data);
```

Манчестерское кодирование реализуется с помощью следующей функции:

```
% manchester.m
function wave=manchester(data)
data(data==0)=-1;
data=upsample(data,2);
data=filter([-1 1],1,data);
wave=maptowave(data);
```

Дифференциальное манчестерское кодирование реализуется с помощью следующей функции:

```
% diffmanc.m
function wave=diffmanc(data)
data=filter(1,[1 1],data);
data=mod(data,2);
wave=manchester(data);
```

Для построения спектра сигнала реализуем следующую функцию (в отдельном файле *calcspectre.m*):

```
% calcspectre.m
function spectre = calcspectre(wave)
Fd = 512; % Частота дискретизации (Гц)
Fd2 = Fd/2;
Fd3 = Fd/2 + 1;
X = fft(wave,Fd);
spectre = X.*conj(X)/Fd;
f = 1000*(0:Fd2)/Fd;
figure()
plot(f,spectre(1:Fd3)), grid;
xlabel('Frequency (Hz)');
```

и в файл *main.m* добавляем следующий код:

```
%% Построение спектра сигнала
mkdir 'spectre';
axis('auto');
data_spectre=[0 1 0 1 0 1 0 1 0 1 0 1 0 1];
% униполярное кодирование
wave=unipolar(data_spectre);
spectre=calcspectre(wave);
title('Unipolar');

% кодирование AMI
wave=ami(data_spectre);
spectre=calcspectre(wave);
title('AMI');

% кодирование NRZ
wave=bipolarnrz(data_spectre);
spectre=calcspectre(wave);
title('Bipolar Non-Return to Zero');

% кодирование RZ
wave=bipolarrz(data_spectre);
spectre=calcspectre(wave);
title('Bipolar Return to Zero');

% манчестерское кодирование
wave=manchester(data_spectre);
spectre=calcspectre(wave);
title('Manchester');
```

```
% дифференциальное манчестерское кодирование
wave=diffmanc(data_spectre);
spectre=calcspectre(wave);
title('Differential Manchester');
```

Запускаем главный скрипт *main.m*. В каталоге *signal* получаем поведение кодированного сигнала (рис. Л.15–Л.20).

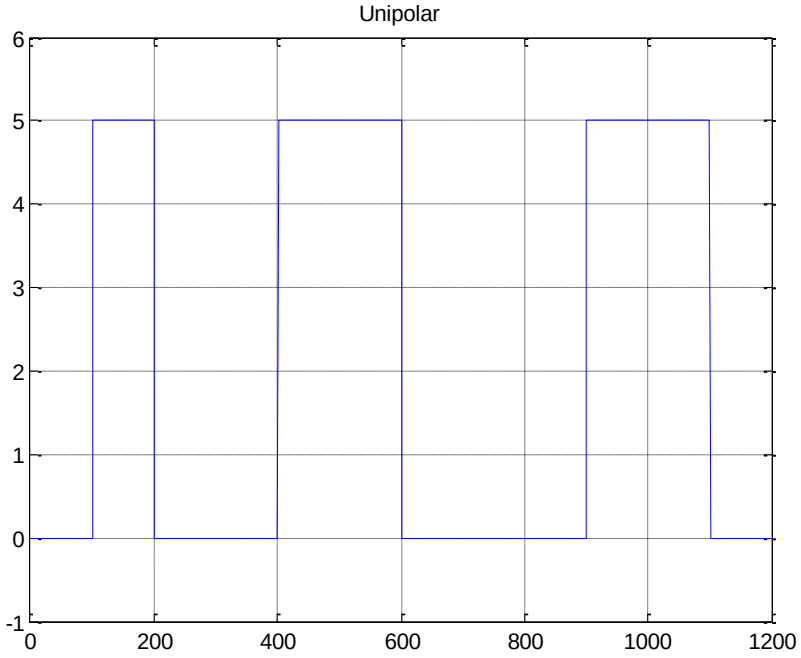

Рис. Л.15. **Униполярное кодирование**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – время, измеряемое в условных единицах

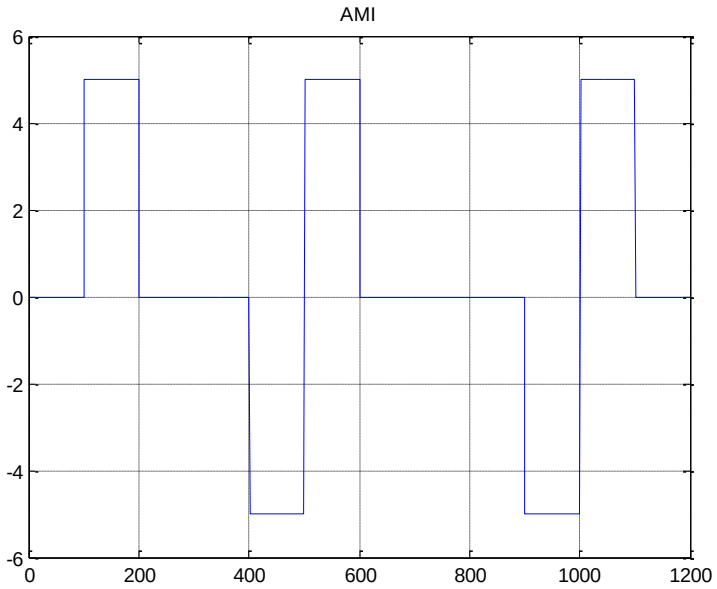

Рис. Л.16. **Кодирование AMI**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – время, измеряемое в условных единицах

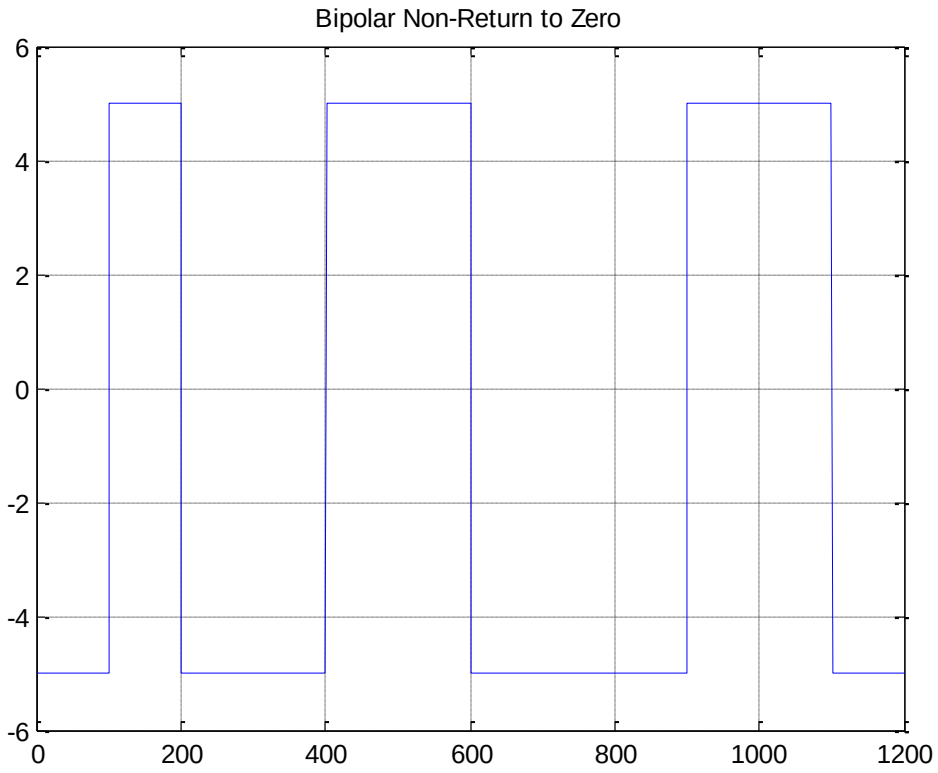

Рис. Л.17. **Кодирование NRZ**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – время, измеряемое в условных единицах

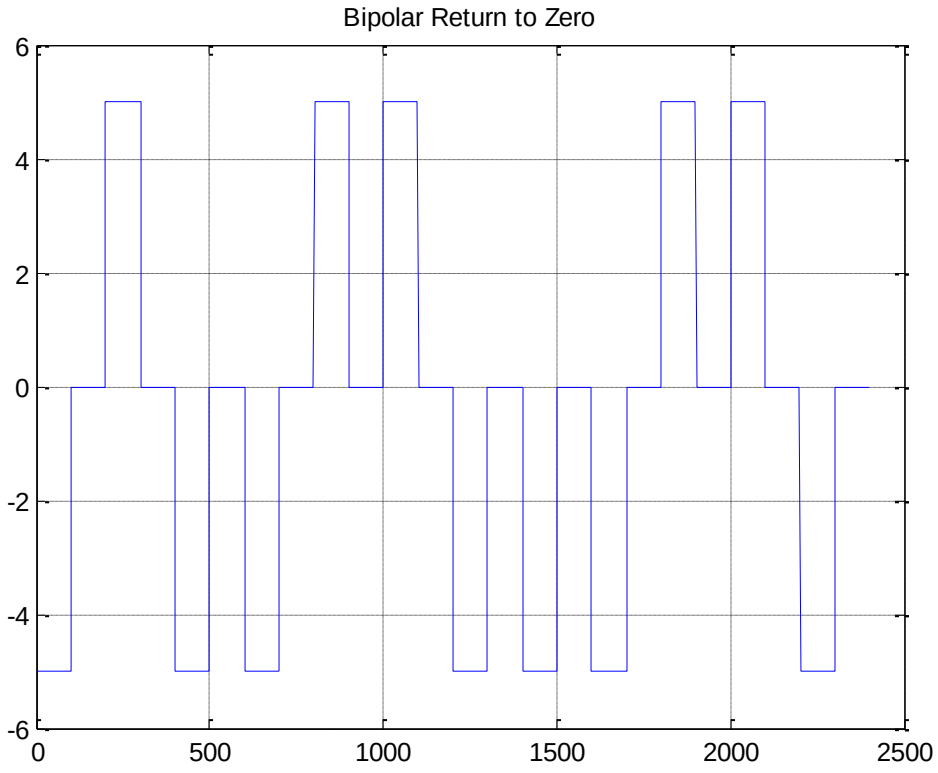

Рис. Л.18. **Кодирование RZ**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – время, измеряемое в условных единицах

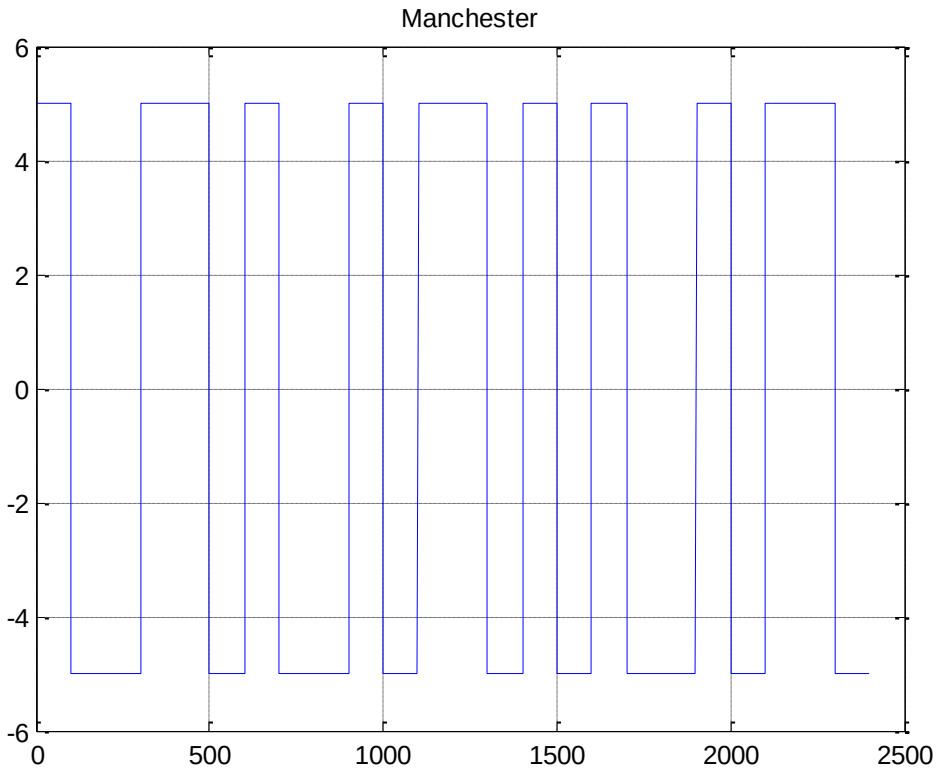

Рис. Л.19. **Манчестерское кодирование**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – время, измеряемое в условных единицах

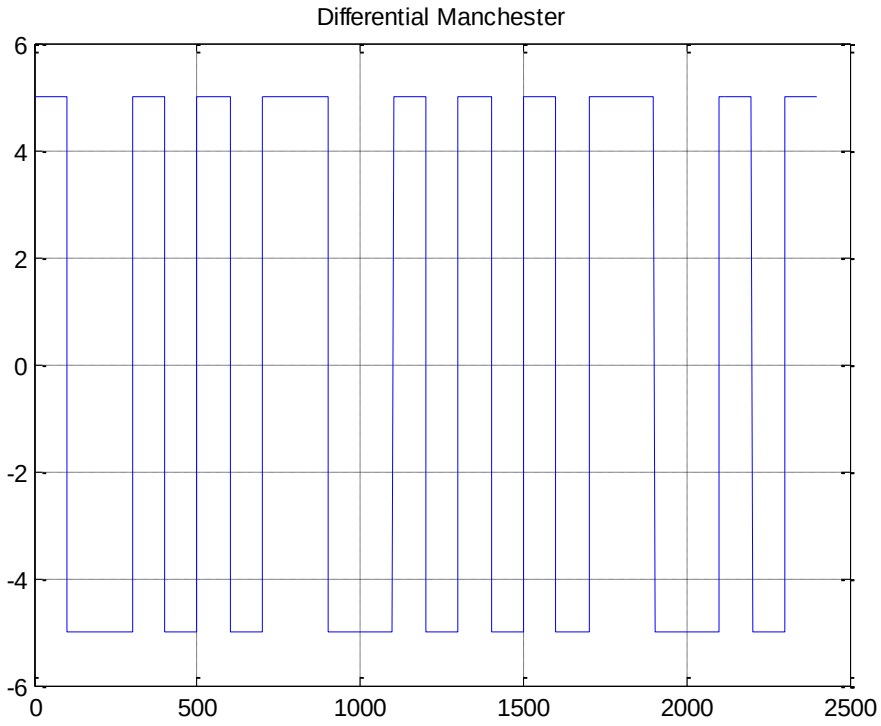

Рис. Л.20. **Дифференциальное манчестерское кодирование**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – время, измеряемое в условных единицах

В каталоге *sync* иллюстрируются свойства самосинхронизации (рис. Л.21–Л.26).

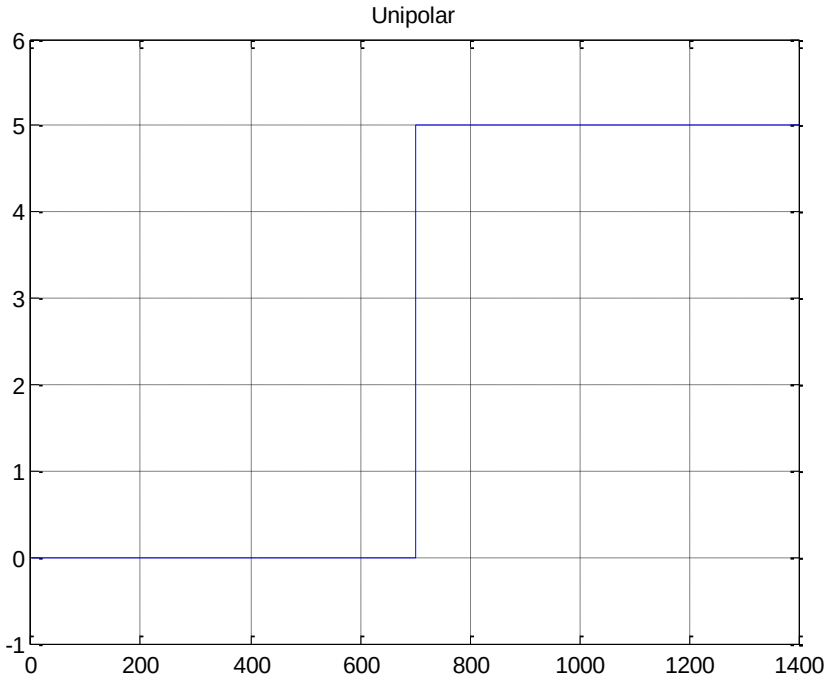

Рис. Л.21. **Униполярное кодирование: нет самосинхронизации**
вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – время, измеряемое в условных единицах

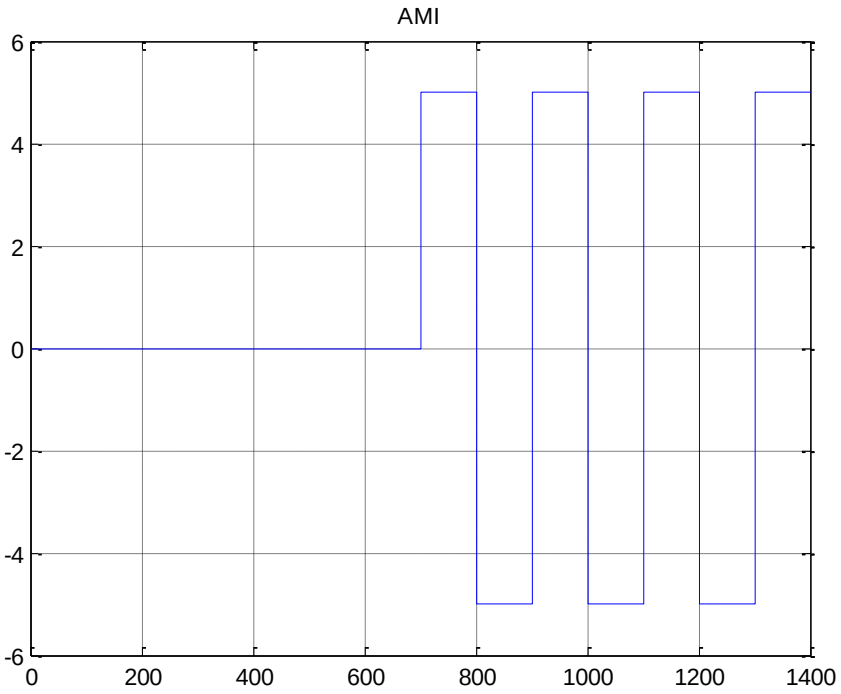

Рис. Л.22. **Кодирование AMI: самосинхронизация при наличии сигнала**
вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – время, измеряемое в условных единицах

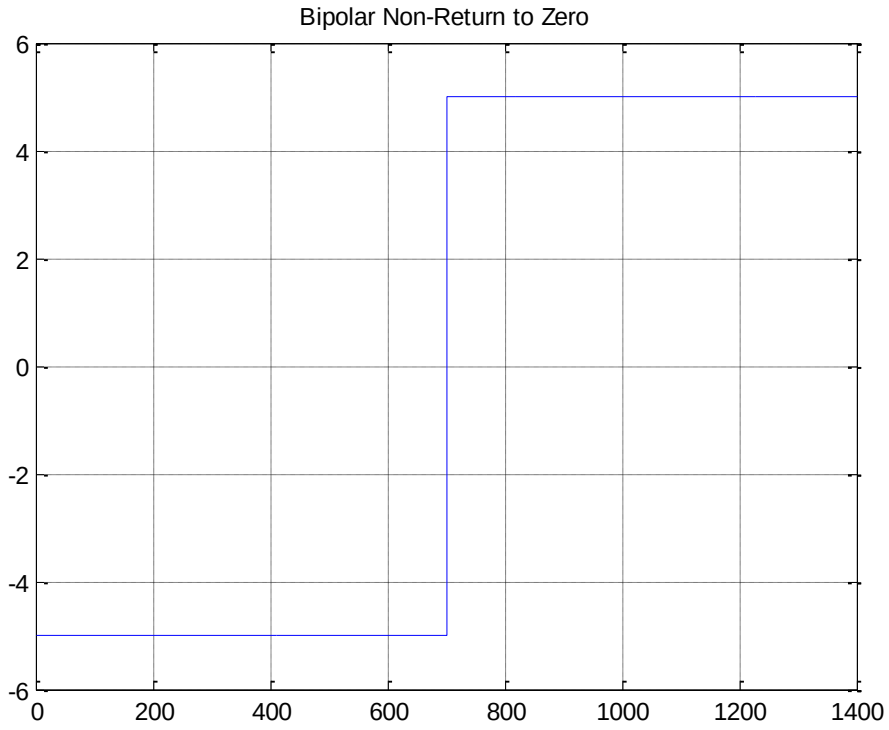

Рис. Л.23. **Кодирование NRZ: нет самосинхронизации**
вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – время, измеряемое в условных единицах

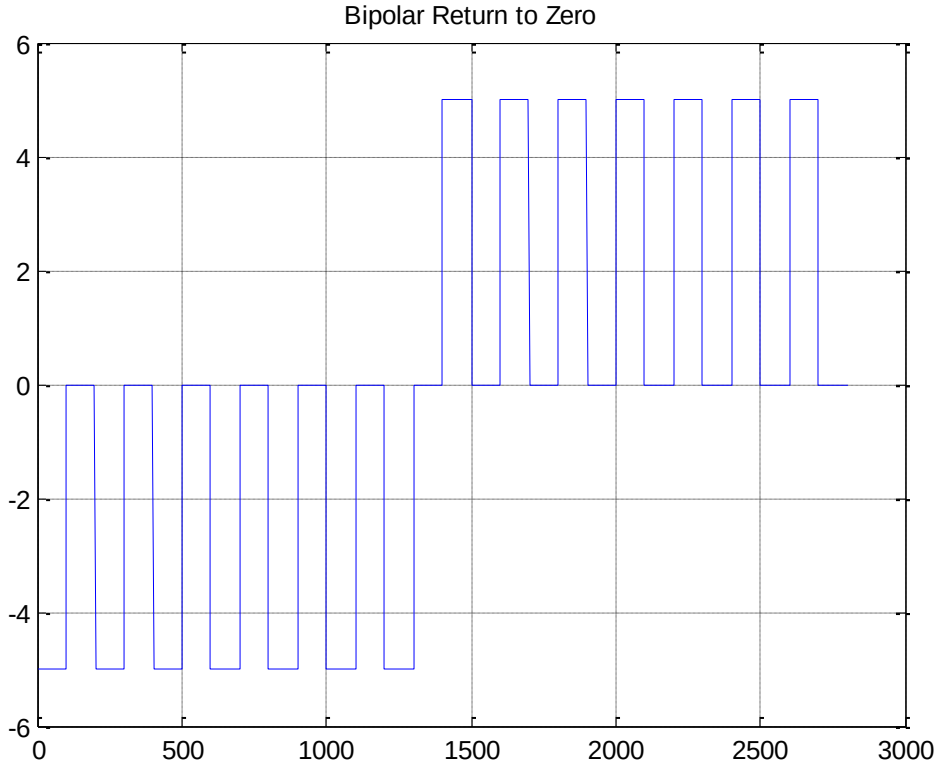

Рис. Л.24. **Кодирование RZ: есть самосинхронизация**
вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах; горизонтальная ось – время, измеряемое в условных единицах

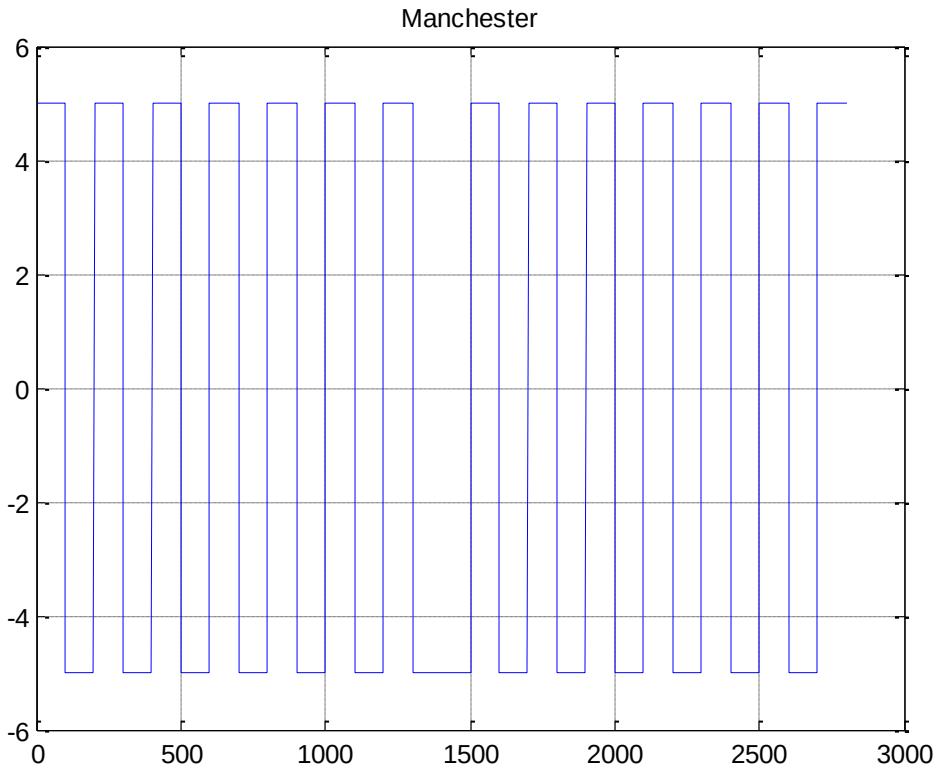

Рис. Л.25. **Манчестерское кодирование: есть самосинхронизация**
вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах;
горизонтальная ось – время, измеряемое в условных единицах

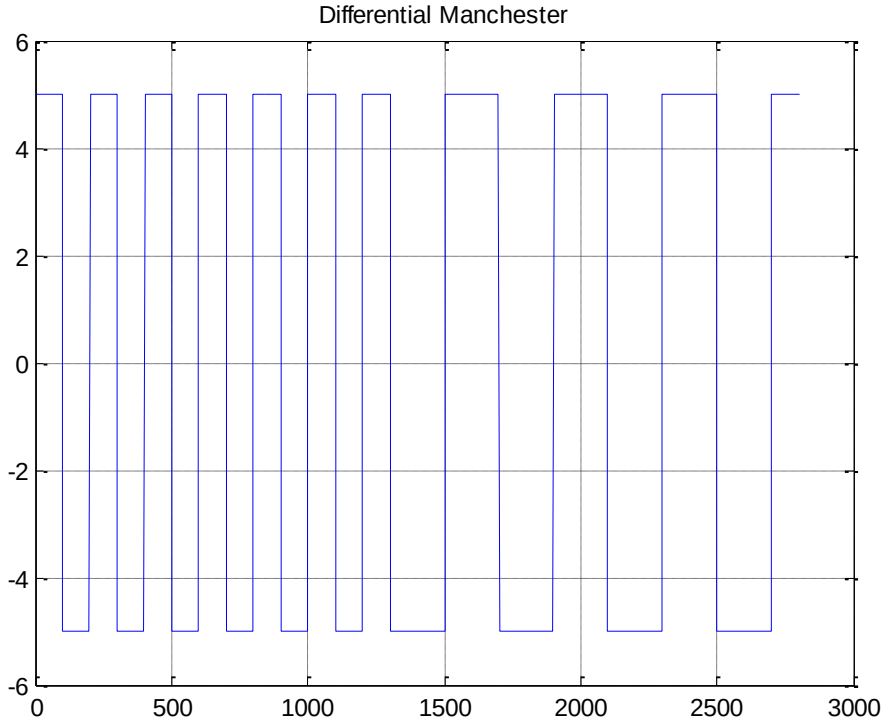

Рис. Л.26. **Дифференциальное манчестерское кодирование: есть самосинхронизация**
вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах;
горизонтальная ось – время, измеряемое в условных единицах

В каталоге *spectre* получаем спектр сигнала (рис. Л.27–Л.32).

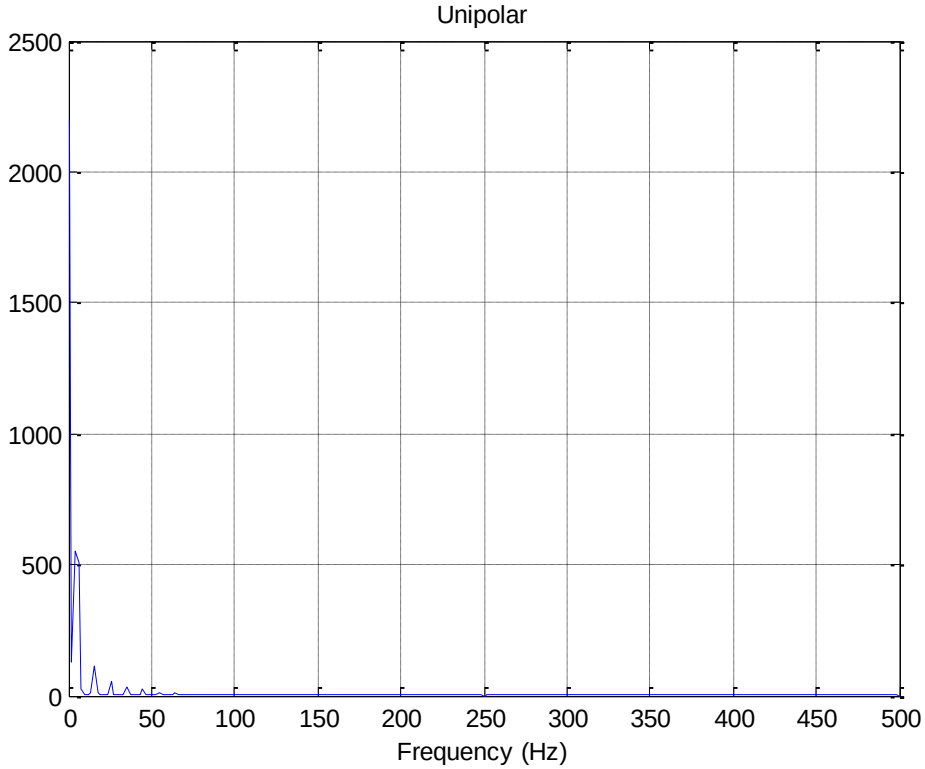

Рис. Л.27. **Униполярное кодирование: спектр сигнала**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах;
горизонтальная ось – частота, измеряемая в герцах(Гц)

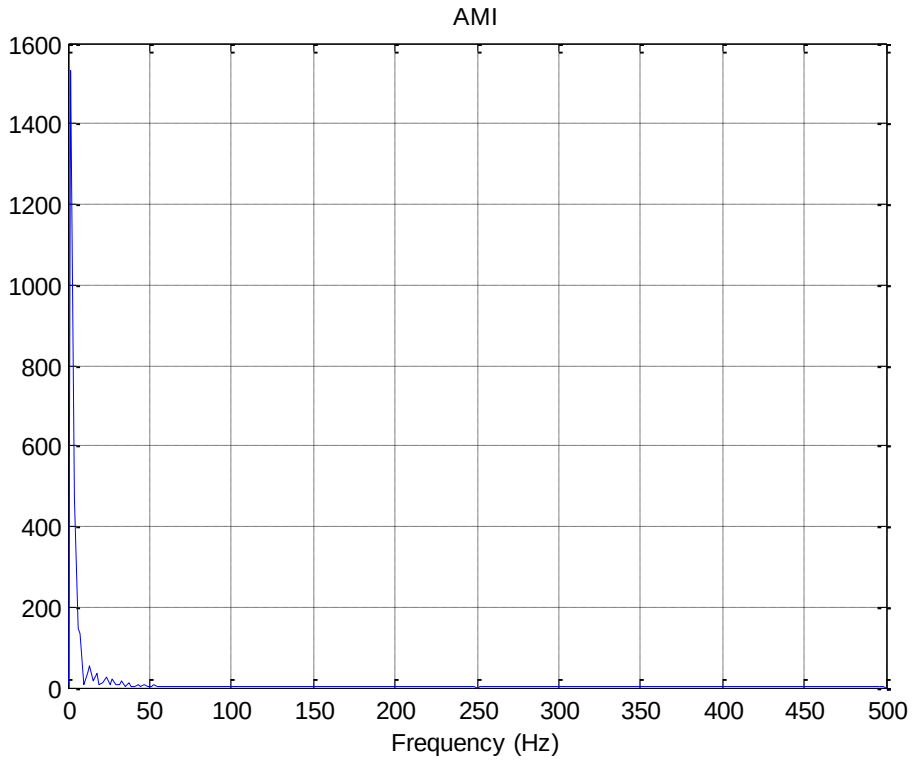

Рис. Л.28. **Кодирование AMI: спектр сигнала**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах;
горизонтальная ось – частота, измеряемая в герцах(Гц)

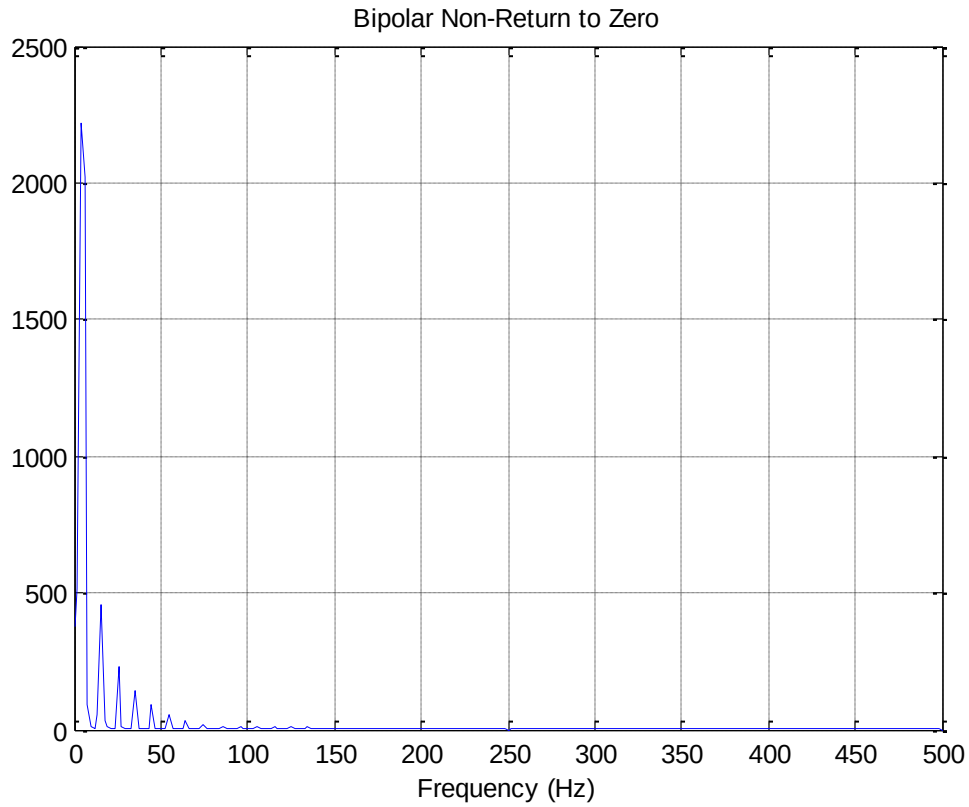

Рис. Л.29. **Кодирование NRZ: спектр сигнала**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах;
горизонтальная ось – частота, измеряемая в герцах(Гц)

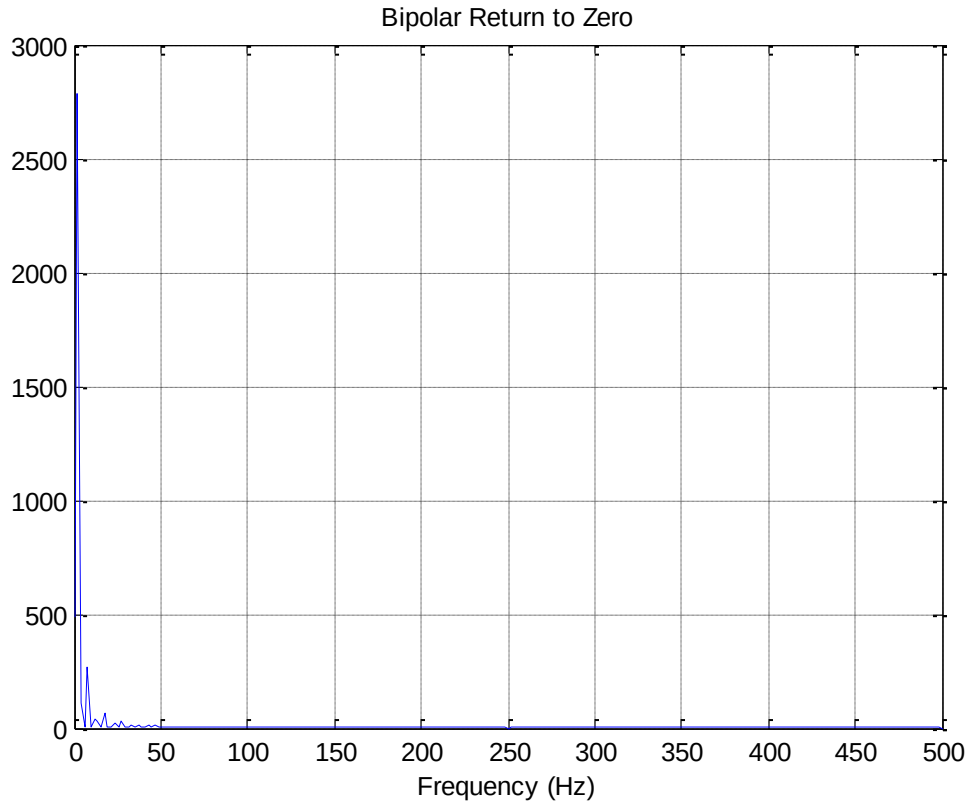

Рис. Л.30. **Кодирование RZ: спектр сигнала**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах;
горизонтальная ось – частота, измеряемая в герцах(Гц)

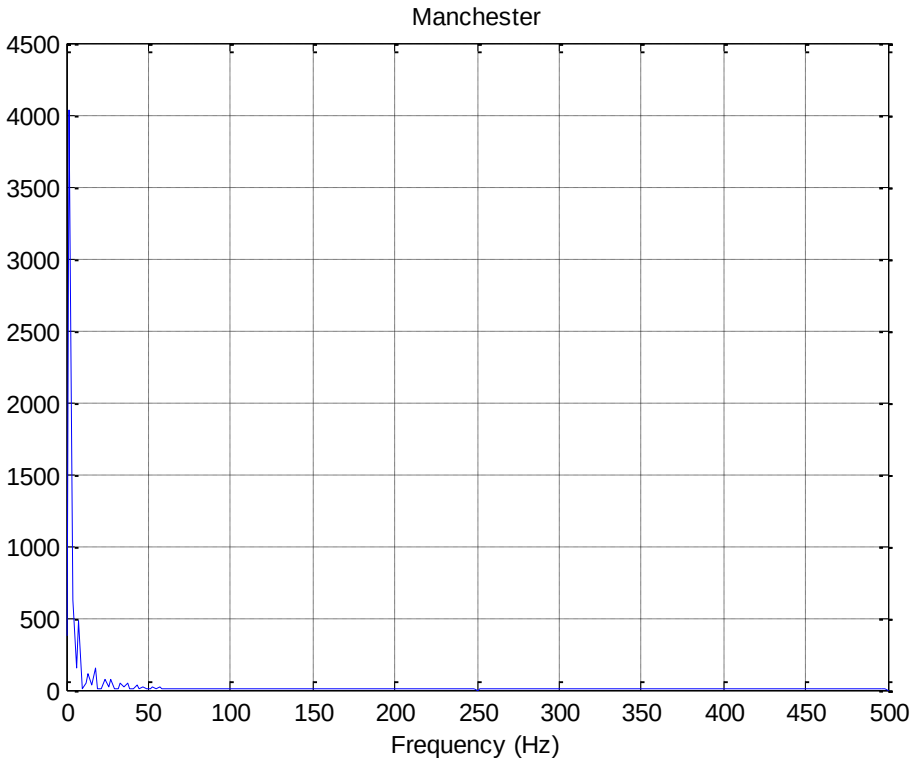

Рис. Л.31. **Манчестерское кодирование: спектр сигнала**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах;
горизонтальная ось – частота, измеряемая в герцах(Гц)

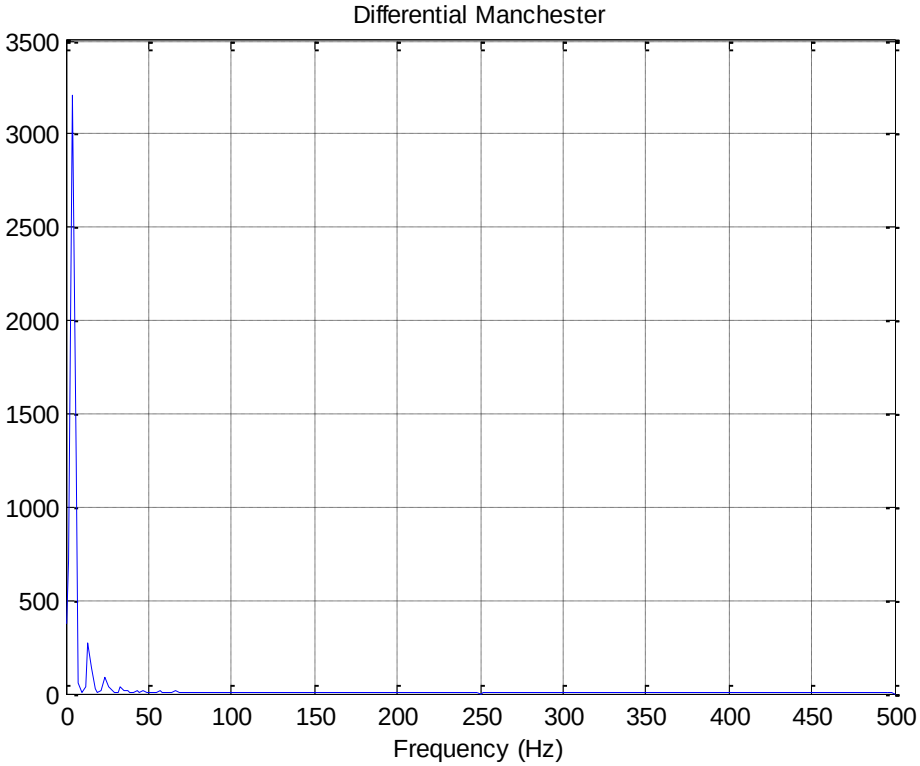

Рис. Л.32. **Дифференциальное манчестерское кодирование: спектр сигнала**

вертикальная ось имеет смысл амплитуды, измеряемой в условных единицах;
горизонтальная ось – частота, измеряемая в герцах(Гц)

Приложение 1

Таблица 1 - код КОИ-7

Восьме-ричный код	Знак	Восьме-ричный код	Знак	Восьме-ричный код	Знак	Восьме-ричный код	Знак
040	Пробел	072	:	121	Q	154	Л
041	!	073	;	122	R	155	М
042	«	074	<	123	S	156	Н
047	‘	075	=	124	T	157	О
050	(	076	>	125	U	160	П
051	)	077	?	126	V	161	Я
052	*	101	A	127	W	162	Р
053	+	102	B	130	X	163	С
054	,	103	C	131	Y	164	Т
055	-	104	D	132	Z	165	У
056	.	105	E	140	Ю	166	Ж
057	/	106	F	141	А	167	В
060	0	107	G	142	Б	170	Ь
061	1	110	H	143	Ц	171	Ы
062	2	111	I	144	Д	172	З
063	3	112	J	145	Е	173	Ш
064	4	113	K	146	Ф	174	Э
065	5	114	L	147	Г	175	Щ
066	6	115	M	150	Х	176	Ч
067	7	116	N	151	И		
070	8	117	O	152	Й		
071	9	120	P	153	К		