МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени Н.Э. БАУМАНА

Учебное пособие

Учебно-методический комплект по дисциплине «Цифровая обработка сигналов»

Конспект лекции «Модуляиция и демодуляция цифровых сигналов»

В.В. Леонидов

МГТУ имени Н.Э. Баумана

1 Введение

Передача информации в современном мире нас окружает повсюду: мобильная связь, интернет, управление воздушным движением и многое другое. Как правило, в качестве «переносчика» информации выступает высокочастотное колебание: 2.4 ГГц и 5 ГГц для Wi-Fi, 2.6 ГГц и другие для 4G, 1030 МГц и 1090 МГц для обмена данными с воздушными судами и т.д.). Эти колебания (частоты) называются **несущими**. Как, используя несущую частоту, закодировать данные? Как передать или получить «0» или «1»? На эти вопросы мы и постараемся ответить.

Процесс изменения одного или нескольких параметров модулируемого несущего сигнала при помощи модулирующего сигнала называется **модуляцией**. В рамках данной лекции мы рассмотрим следующие виды модуляции:

- ASK (Amplitude Shift Keying) – амплитудная манипуляция
- BPSK, QPSK (Binary Phase Shift Keying, Quadrature Phase Shift Keying) – фазовая манипуляция
- QASK (Quadrature Amplitude-Shift Keying) – квадратурная манипуляция
- FSK (Frequency Shift Keying), MSK (Minimum Shift Keying) – частотная манипуляция

2 Амплитудная манипуляция (ASK)

Амплитудная манипуляция – манипуляция, при которой скачкообразно изменяется амплитуда несущего сигнала в зависимости от закодированного сообщения.

Рассмотрим пример. Создадим сигнал `fc`, который из себя будет представлять несущую частотой *500 Гц*, добавим в неё немного шума и промодулируем кодовой последовательностью (сообщением), заданной в массиве `code`. Логическому «нулю» будет соответствовать амплитуда *0.1*, логической «единице» – *1*:

Листинг 1 – ASK-манипуляция, часть 1

```
clear, clc, close all

fs = 10000;
ts = -0.1 : 1/fs : 0.1-1/fs;
N = length(ts);

%% несущая частота
fc = cos(2*pi*500*ts);
fc = awgn(fc,30);

%% модулирующий сигнал
% длина одного бита в отсчётах
```

```
n_for_bit = 200;
% кодируемая последовательность
code = [1 0.1 0.1 1 1 0.1 1 0.1 1 1];

% формируем модулирующий сигнал
fm = zeros(1,N);
for i=1:length(code)
    for j=n_for_bit*(i-1)+1:n_for_bit*i
        fm(j) = code(i);
    end
end

%% амплитудная модуляция
x = fc.*fm;

plot(ts,x,'LineWidth',0.5), grid on, hold on
plot(ts,fm,'LineWidth',2), grid on
title ('ASK модуляция')
xlabel('Время'), ylabel('Амплитуда')
legend({'Модулированный сигнал';'Модулирующий сигнал'})
```

Результат выполнения скрипта показан на рисунке 1.

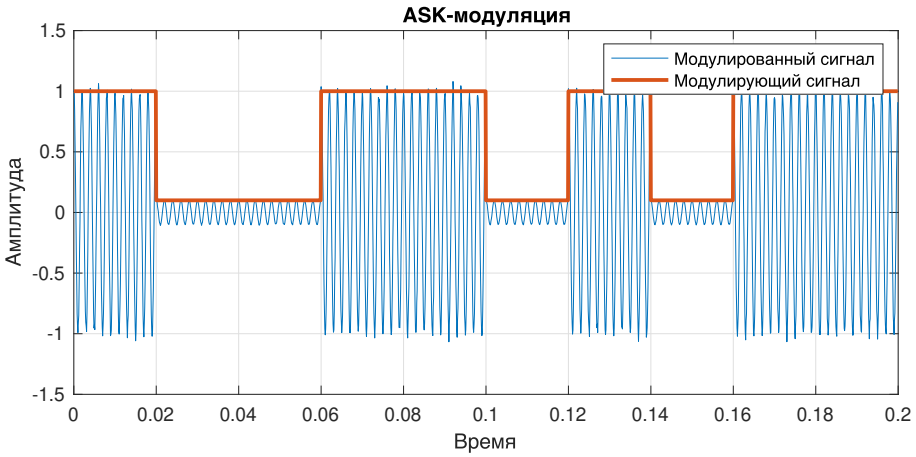

Рис. 1 – Амплитудная манипуляция

Из рисунка видно, что амплитуда несущего (модулированного) сигнала – синий график – повторяет форму модулирующего сигнала – оранжевый график.

Теперь давайте сделаем следующую вещь: на комплексной плоскости отметим точки, соответствующие значениям, которые принимает наш закодированный сигнал. В Matlab это можно сделать с помощью функции

`scatterplot`, указав в качестве входных параметров аналитический сигнал (мы его получим с помощью преобразования Гильберта), количество отсчётов на бит (у нас за это отвечает переменная `n_for_bit`) и смещение от начала сигнала, с которым будут считываться биты (мы хотим считывать биты в их середине, поэтому смещение зададим `n_for_bit/2`):

Листинг 2 – ASK-манипуляция, часть 2

```
33 %% сигнальное созвездие
34 scatterplot(hilbert(x),n_for_bit,round(n_for_bit/2)), grid on
```

Посмотрим результат (рисунок 2). Т.к. мнимая часть равна нулю, мы видим два скопления точек вдоль

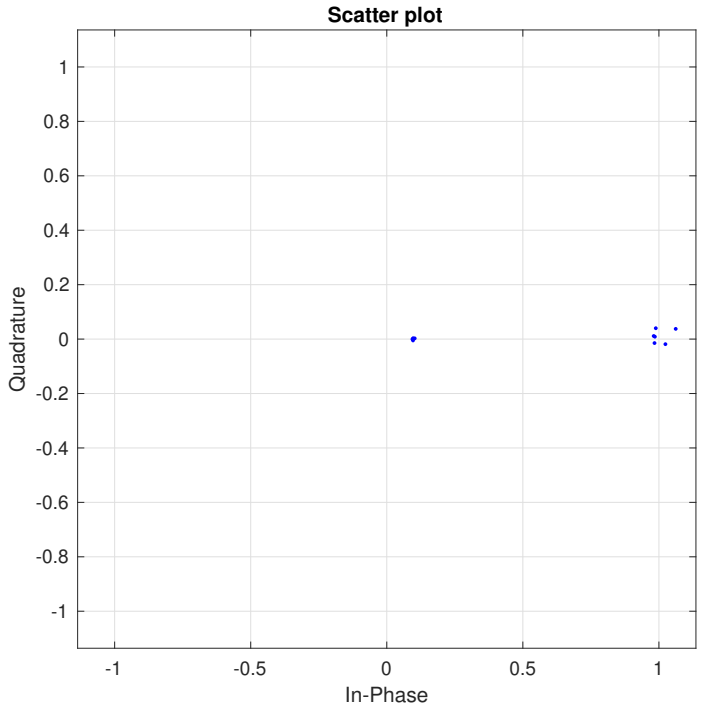

Рис. 2 – Сигнальное созвездие ASK-модулированного сигнала

действительной оси: одно вокруг значения *0.1*, второе – вокруг *1*. Этот график называется **сигнальное созвездие**. Он часто используется при анализе характеристик модулированных сигналов, т.к. показывает распределение мгновенного значения сигнала на комплексной плоскости в момент его считывания.

Итак, модулированный сигнал есть – попробуем его обратно демодулировать. Наша задача сводится к нахождению огибающей, а как мы выяснили на прошлой лекции – это легко сделать с помощью преобразования Гильберта. Затем создадим что-то вроде цифрового компаратора, с помощью которого зашумлённую огибающую превратим в прямоугольный цифровой сигнал. Дополним наш код:

Листинг 3 – ASK-манипуляция, часть 3

```
35 %% амплитудная демодуляция
```

```
% преобразование Гильберта
h = hilbert(x);

figure
subplot(2,1,1)
plot(ts,x,'LineWidth',0.5), grid on, hold on
plot(ts,abs(h),'LineWidth',2), grid on
title ('ASK демодуляция')
xlabel('Время'), ylabel('Амплитуда')
legend({'Модулированный сигнал';'Демодулированный сигнал'})

% компаратор
dem = zeros(1,length(h));
for i=1:length(h)
    if abs(h(i))>=0.5
        dem(i) = 1;
    else
        dem(i) = 0;
    end
end

subplot(2,1,2)
plot(ts,dem,'LineWidth',2), grid on
title ('Код демодулированного сигнала')
xlabel('Время'), ylabel('Амплитуда')
```

Результат выполнения кода показан на рисунке 3. Как видим, демодулированный сигнал повторяет битовую последовательность, которую мы закодировали изначально в массиве `code`, а значит мы всё сделали правильно.

Ещё одним типом амплитудной манипуляции является **OOK** – *On-O Keying*. Основное отличие от ASK – логический «ноль» кодируется амплитудой, равной нулю, «единице» соответствует номинальное значение амплитуды сигнала. Если в листинге 1 в массиве `code` вместо всех значений `0.1` запишем `0`, то как раз получим OOK-манипуляцию.

3 Фазовая манипуляция (PSK)

Фазовая манипуляция – манипуляция, при которой скачкообразно изменяется фаза несущего сигнала в зависимости от закодированного сообщения.

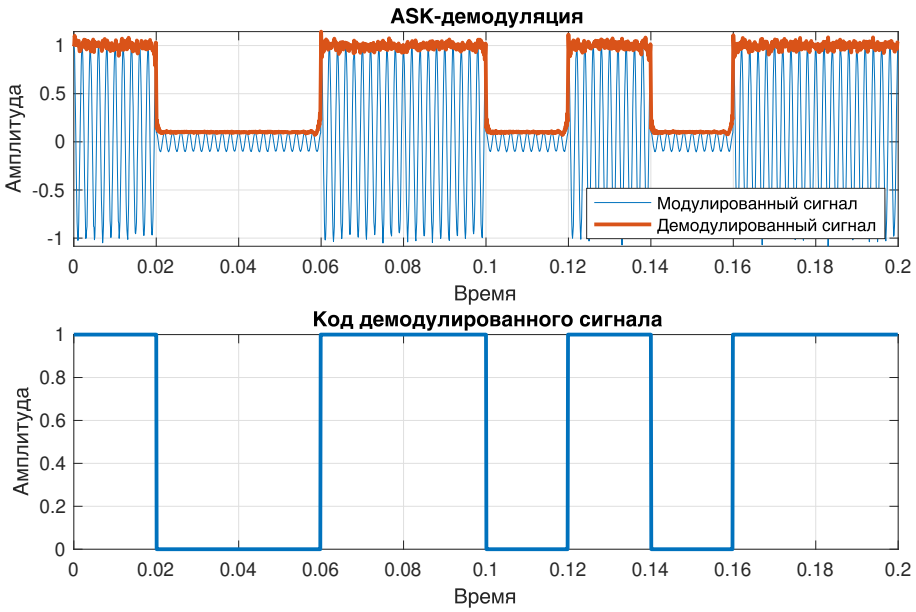

Рис. 3 – Демодуляция ASK-модулированного сигнала

Начнём с самого простого вида – BPSK – когда у нас всего два значения фазы – `0` и `180` .

Создадим скрипт, реализующий фазовую манипуляцию зашумлённой несущей частоты `fc`, равной *250 Гц*. Кодовую последовательность по аналогии с первым примером будем задавать в массиве `code`. Перескок фазы будет осуществляться умножением синусоидального сигнала на *+1* или *-1*:

Листинг 4 – BPSK-манипуляция, часть 1

```
clear, clc, close all

fs = 10000;
ts = 0 : 1/fs : 0.2-1/fs;
N = length(ts);

%% несущая частота
fc = cos(2*pi*250*ts);
fc = awgn(fc,30);

%% модулирующий сигнал
% длина одного бита в отсчётах
n_for_bit = 200;
% кодируемая последовательность
code = [1 -1 -1 1 1 -1 1 -1 1 1];

```

```
% формируем модулирующий сигнал
fm = zeros(1,N);
for i=1:length(code)
    for j=n_for_bit*(i-1)+1:n_for_bit*i
        fm(j) = code(i);
    end
end

%% фазовая манипуляция (BPSK)
x = fc.*fm;

plot(ts,x,'LineWidth',0.5), grid on, hold on
plot(ts,fm,'LineWidth',2), grid on
title ('BPSK модуляция')
xlabel('Время'), ylabel('Амплитуда')
legend({'Модулированный сигнал';'Модулирующий сигнал'})
```

Смотрим, что получилось (рисунок 4). Каждое изменение логического уровня модулирующего сигнала при-

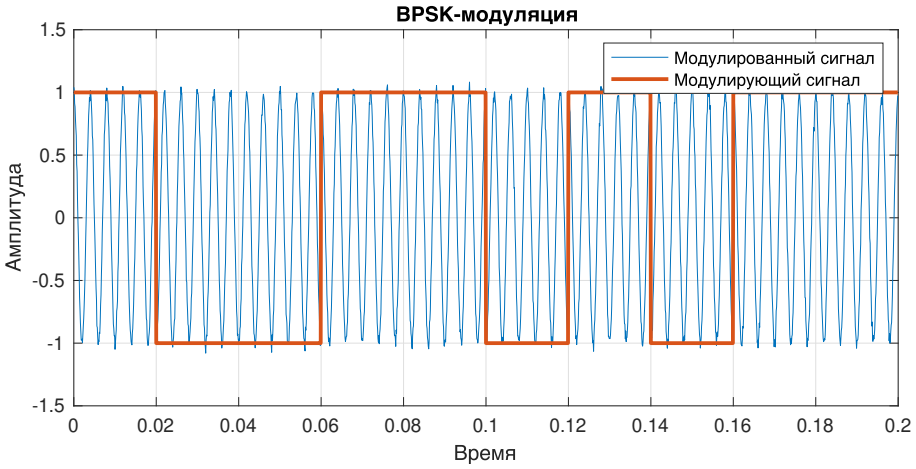

Рис. 4 – BPSK-манипуляция

водит к скачкообразному изменению фазы несущей частоты на 180 , а это то, что нужно.

Теперь дополним код по аналогии с предыдущим примером и построим сигнальное созвездие:

Листинг 5 – BPSK-манипуляция, часть 2

```
%% сигнальное созвездие
scatterplot(hilbert(x),n_for_bit,round(n_for_bit/2)), grid on
```

Сигнальное созвездие BPSK-сигнала показано на рисунке 6. Мы имеем два скопления точек: вокруг (-1,0)

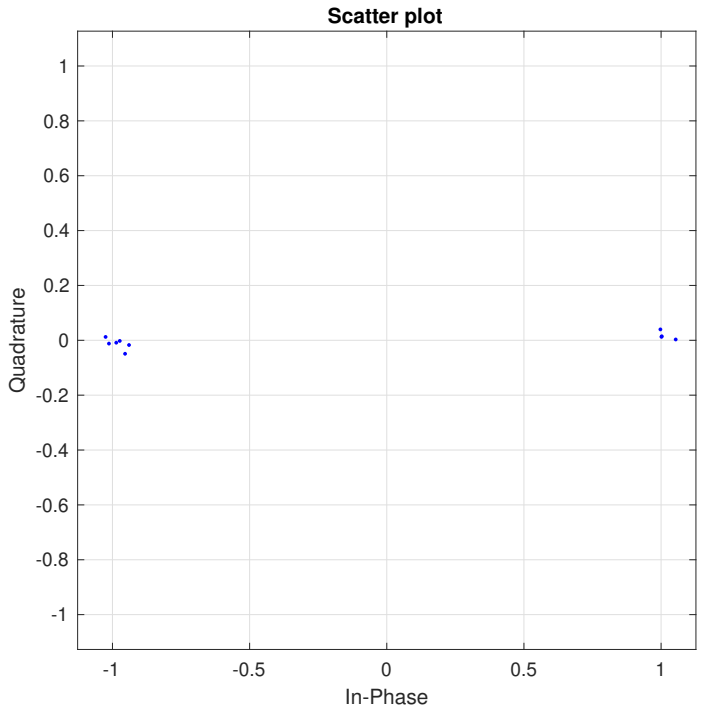

Рис. 5 – Сигнальное созвездие BPSK-сигнала

и (1,0). Эти точки как раз соответствуют повороту вектора единичной длины на `0` и `180` вокруг начала координат. А значит, модуляция работает корректно.

Теперь наша задача – демодулировать сгенерированный сигнал. Для этого умножим его на несущую частоту `fc`, а затем применим к полученному сигналу ФНЧ с полосой пропускания *100 Гц*, сгенерированный с помощью filterDesigner:

Листинг 6 – BPSK-манипуляция, часть 3

```
35 %% BPSK демодуляция
36 % умножаем сигнал на несущую
37 y = x.*fc;
38
39 % применяем ФНЧ, чтобы убрать несущую
40 fltr = bpsk_fir;
41 z = filter(fltr.Numerator,1,y);
42
43 figure;
44 subplot(2,1,1)
45 plot(ts,y), grid on
46 title('Модулированный сигнал, умноженный на несущую')
```

```
47 xlabel('Время'), ylabel('Амплитуда')
48 subplot(2,1,2)
49 plot(ts,z), grid on
50 title('Тот же сигнал, прошедший через ФНЧ')
51 xlabel('Время'), ylabel('Амплитуда')
```

Результаты наблюдаем на рисунке 6. На верхнем графике наблюдаем синусоиды с частотой `2*fc`, имеющие

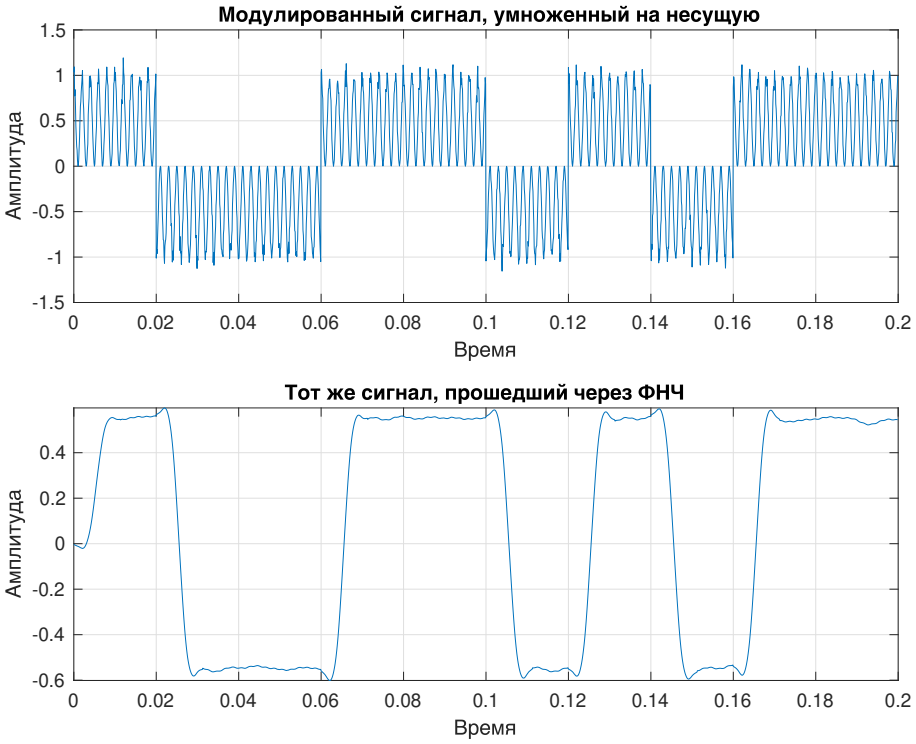

Рис. 6 – Обработка BPSK-модулированного сигнала

разные постоянные составляющие в зависимости от значения фазы модулированного сигнала. Для того, чтобы получить огибающую, мы воспользовались ФНЧ и получили нижний график. Теперь, чтобы преобразовать данный сигнал в цифровой код, разработаем простейший цифровой компаратор с пороговым значением *0.1* и построим результирующие графики:

Листинг 7 – BPSK-манипуляция, часть 4

```
52 % компаратор
53 dem = zeros(1,length(z));
54 for i=1:length(z)
55     if z(i)>=0.1
56         dem(i) = 1;
57     else
```

```
        dem(i) = 0;
    end
end

figure
plot(ts,x,'LineWidth',0.5), grid on, hold on
plot(ts,dem,'LineWidth',2), grid on
xlabel('Время'), ylabel('Амплитуда')
title ('BPSK демодуляция')
legend({'Модулированный сигнал';'Демодулированный сигнал'})
```

Демодулированный сигнал на фоне модулированного сигнала представлен на рисунке 7. Как видим, он повто-

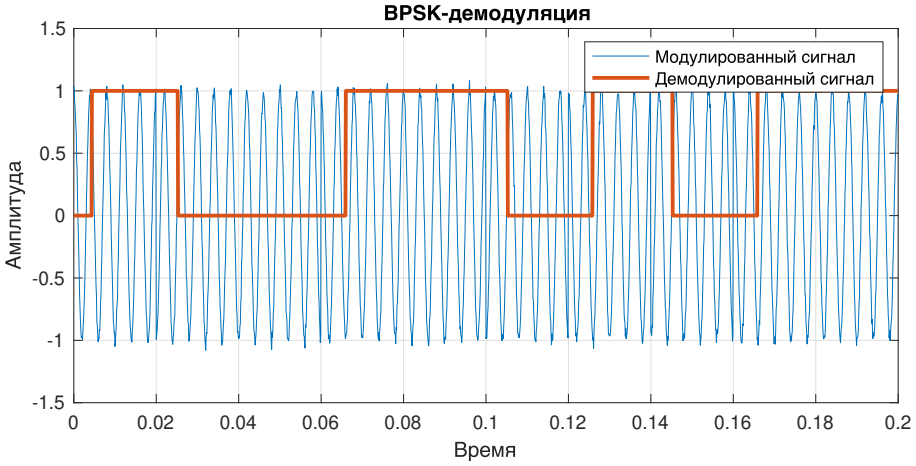

Рис. 7 – Демодуляция BPSK-модулированного сигнала

ряет форму модулирующего сигнала (рисунок 4), заданного в начале листинга, но имеет временную задержку, возникающую в КИХ-ФНЧ.

В рассмотренном примере модулированный сигнал мог принимать два значения фазы: `0` и `180` , а значит в один момент времени (такт) можно закодировать только один бит информации – «0» или «1». Если добавить ещё два изменения фазы (`90` и `270`), а затем повернуть полученную диаграмму на `45` против часовой стрелки, получится **QPSK-манипуляция**, или квадратурная фазовая манипуляция. В этом случае мы за один такт сможем передавать сразу два бита информации! Посмотрим, как в этом случае будет выглядеть сигнальное созвездие. Разработаем скрипт, который формирует массив случайных данных `data`, которые затем преобразуем в массив значений PSK-сигнала с помощью функции `pskmod`, в качестве второго параметра которой будет порядок модуляции (для QPSK это `M = 4`). В качестве третьего параметра данной функции необходимо задать смещение нулевого значения фазы, в нашем случае, как было сказано выше, это $\pi/4$ (тогда, в идеальном случае, в каждом квадранте комплексной плоскости будет по одной точке созвездия). Далее добавим немного шума в этот сигнал, чтобы имитировать реальные условия передачи данных и построим сигнальное созвездие:

Листинг 8 – Построение сигнального созвездия для QPSK-манипуляции

```
clear, clc, close all

M = 4;
data = randi([0 M-1],1000,1);

txSig = pskmod(data, M, pi/M);
rxSig = awgn(txSig,20);

scatterplot(rxSig), grid on
```

Полученное сигнальное созвездие представлено на рисунке 8. Мы видим скопление точек вокруг значений

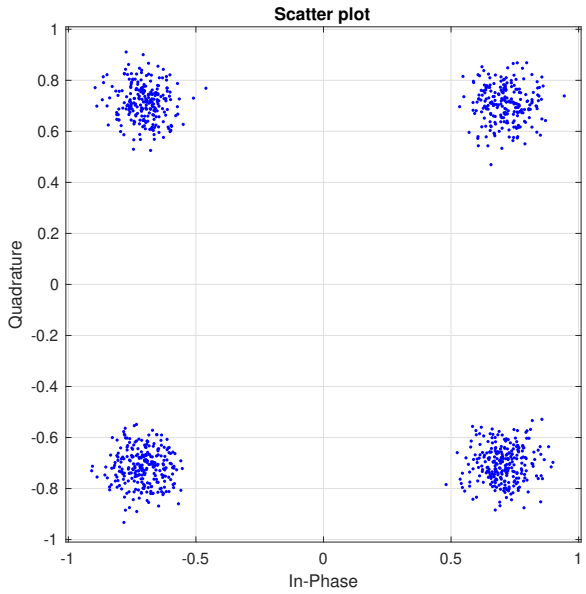

Рис. 8 – Сигнальное созвездие сигнала с QPSK

$\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$, $\left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$, $\left(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right)$, $\left(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right)$, каждая из которых соответствует положению единичного вектора, начало которого соответствует началу координат, при его повороте с шагом 90 . Соответствие фазы и передаваемой информации будет иметь следующий вид:

- 45 – "11"
- 135 – "01"
- 225 – "00"
- 315 – "10"

Это значит, что при при той же самой символьной скорости сигнал с QPSK передаёт в 2 раза больше информации, чем сигнал с BPSK.

Если присвоить `M = 8` и убрать из строки 5 параметр `pi/M`, получим 8-PSK-манипуляцию. Сигнальное созвездие примет вид, как на рисунке 9. Здесь мы можем передавать за один такт сразу 3 бита. Однако,

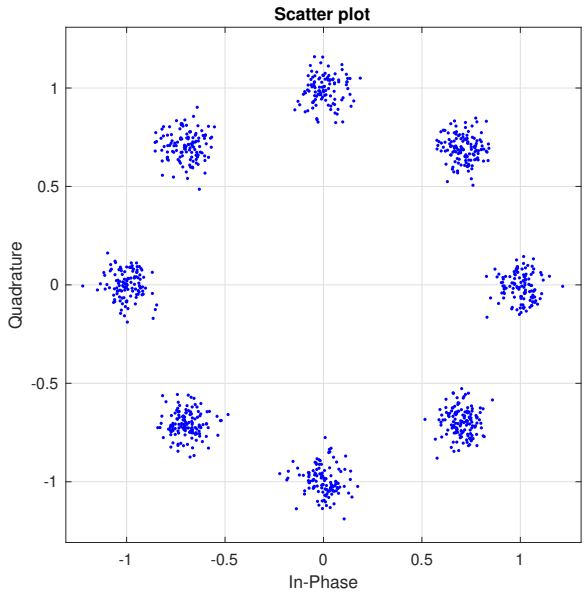

Рис. 9 – Сигнальное созвездие сигнала с 8-PSK

с увеличением порядка модуляции `M`, точки созвездия располагаются всё ближе и ближе друг другу, что может привести к ошибкам декодирования такого сигнала, если он сильно зашумлён. Можете поиграться со значениями `M` и параметром `snr` функции `awgn`, чтобы убедиться в этом на примере.

4 Квадратурная манипуляция (QASK)

QASK (частный случай QAM – *Quadrature Amplitude Modulation*) – это вид манипуляции, при которой скачкообразно изменяется как амплитуда, так и фаза несущего сигнала, что позволяет за один такт (отсчёт) передать ещё больше информации, чем в рассмотренных ранее видах манипуляции. Можно сказать, что QASK – это комбинация ASK и PSK.

По традиции, сразу начнём с примера. Создадим несущую с частотой *1кГц*, помимо этого создадим массив данных `data`, который будет содержать случайные числа. Зададим порядок модуляции `M=16`, это значит, что за один такт будем передавать число от 0 до 15, или 4 бита. Один элемент – один такт передачи, количество элементов – 50. Затем создадим массив отсчётов QASK на базе этих данных с помощью функции `qammod` и построим сигнальное созвездие из полученного набора данных.

Листинг 9 – QASK-манипуляция, часть 1

```
1 clear, clc, close all
```

```
2
3  fs = 10000;
4  ts = 0 : 1/fs : 0.2-1/fs;
5  N = length(ts);
6
7  %% несущая частота
8  fc = 1000;
9
10 %% Модуляция
11 M = 16;                          % порядок QASK модуляции
12 Nd = 50;                         % размер данных
13 bit_size = N/Nd;                 % количество отсчётов на бит
14 data = randi([0 M-1],Nd,1);      % случайные данные
15
16 % формируем массив комплексных чисел
17 qdata = qammod(data, M);
18 % сигнальное созвездие
19 sc = scatterplot(qdata); grid on
20 obj = findobj(sc.Children(1).Children, 'type', 'line');
21 set(obj, 'MarkerSize', 20)
```

Результат представлен на рисунке 10. Это созвездие состоит из 16 групп точек, а значит сформированный

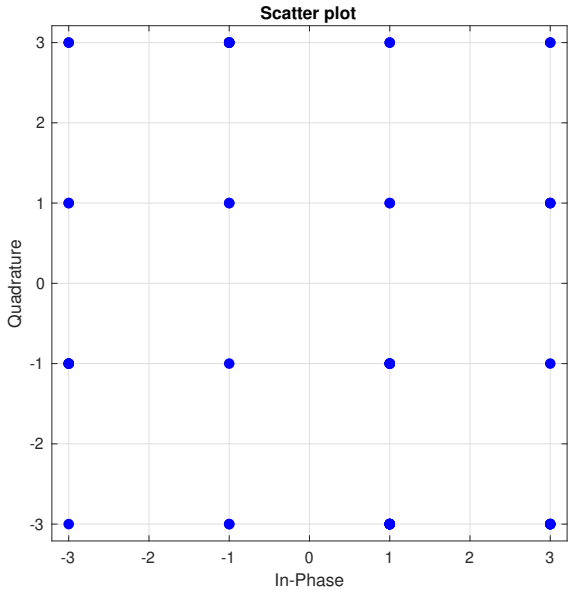

Рис. 10 – Сигнальное созвездие сигнала с 16-QASK

сигнал принимает все возможные значения для QASK-манипуляции 16 порядка.

Теперь «поместим» эти данные на несущую частоту. Растянем массив данных до того же количества отсчётов, что и в сигнале несущей частоты (каждый бит повторим `bit_size` раз) и получим массив `qmod`. Поэлементно умножим действительные части данного массива на косинус с частотой `fc`, а мнимые – на синус той же частоты. Просуммируем полученные сигналы (`i` и `q` соответственно), в результате чего получим модулированный сигнал `y`, который готов для передачи. Добавим в него шум для имитации электромагнитных помех и построим графики.

Листинг 10 – QASK-манипуляция, часть 2

```
22 qmod = repelem(qdata,bit_size).';  % размножаем массив данных до колва- отсчётов
       несущей
23
24 % формируем синусный и косинусный сигналы
25 i = real(qmod).*cos(2*pi*fc*ts);
26 q = imag(qmod).*sin(2*pi*fc*ts);
27
28 % суммируем i и q, получаем сигнал, готовый к передаче
29 y = i+q;
30
31 % добавляем шум
32 y = awgn(y,20);
33
34 figure
35 subplot(3,1,1)
36 plot(real(qmod)), grid on
37 title('Действительная часть сигнала')
38 xlabel('Время'), ylabel('Real')
39 subplot(3,1,2)
40 plot(imag(qmod)), grid on
41 title('Мнимая часть сигнала')
42 xlabel('Время'), ylabel('Imag')
43 subplot(3,1,3)
44 plot(y), grid on
45 title('Модулированный сигнал')
46 xlabel('Время'), ylabel('Амплитуда')
```

Действительная и мнимая части сигнала, а также сам сигнал `y` можно увидеть на рисунке 11. Видно, что в процессе передачи данных изменяется как амплитуда, так и фаза несущей.

Теперь решим обратную задачу: демодулируем сигнал из рисунка 11 (нижний график). Для этого обратно

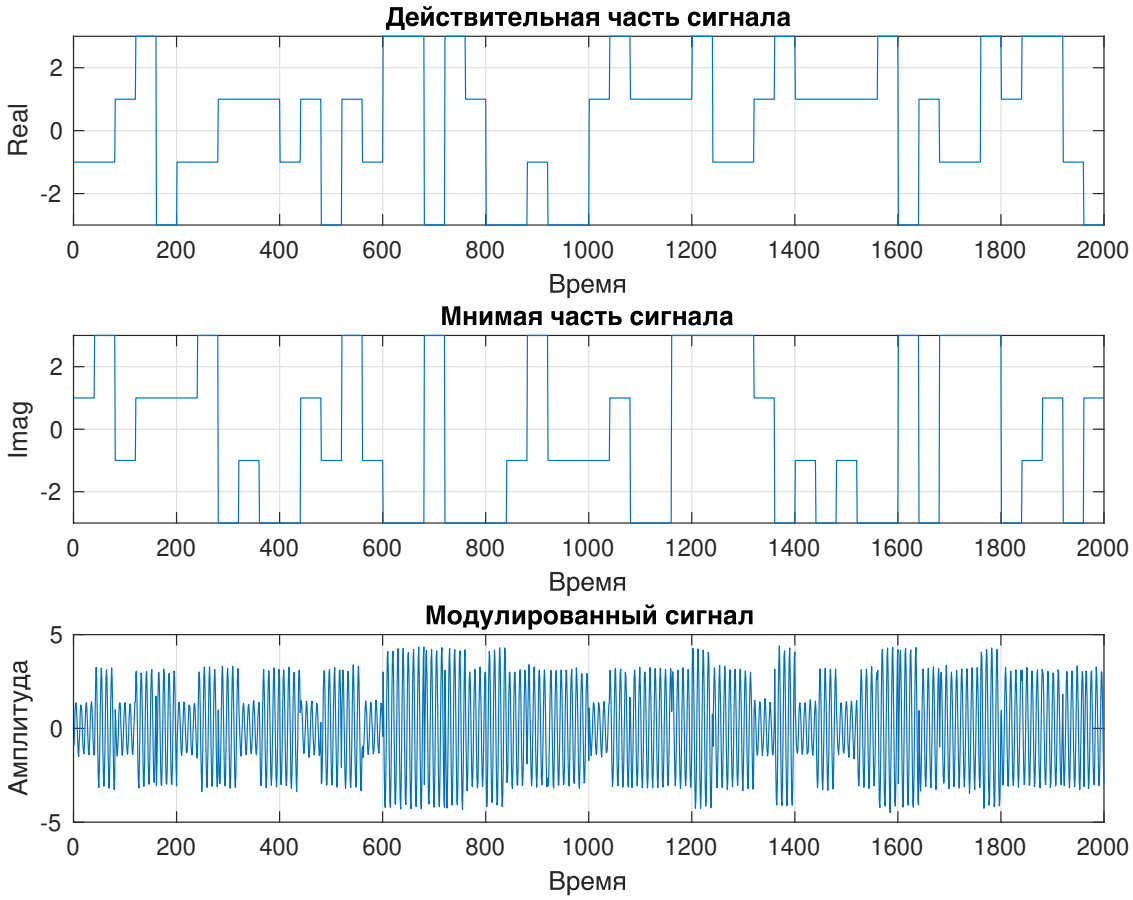

Рис. 11 – Синфазная, квадратурная составляющие и QASK-модулированный сигнал

выделим из него синфазную `io` и квадратурную `qo` составляющие умножением на косинус и синус несущей частоты соответственно. Чтобы убрать высокочастотную составляющую, как и в BPSK, применим ФНЧ, в результате чего получим сигналы `iof` и `qof`, графики которых затем и построим (рисунок 12).

Листинг 11 – QASK-манипуляция, часть 3

```
47 %% Демодуляция
48
49 io = 2*y.*cos(2*pi*fc*ts);
50 qo = 2*y.*sin(2*pi*fc*ts);
51
52 % применяем ФНЧ, чтобы убрать несущую и умножаем на 2
53 fltr = qam_fir;
54 iof = round(conv(fltr.Numerator,io));
55 qof = round(conv(fltr.Numerator,qo));
56
```

```
figure
subplot(2,1,1)
plot(iof), grid on
title('Синфазная составляющая демодулированного сигнала после ФНЧ')
subplot(2,1,2)
plot(qof), grid on
title('Квадратурная составляющая демодулированного сигнала после ФНЧ')
```

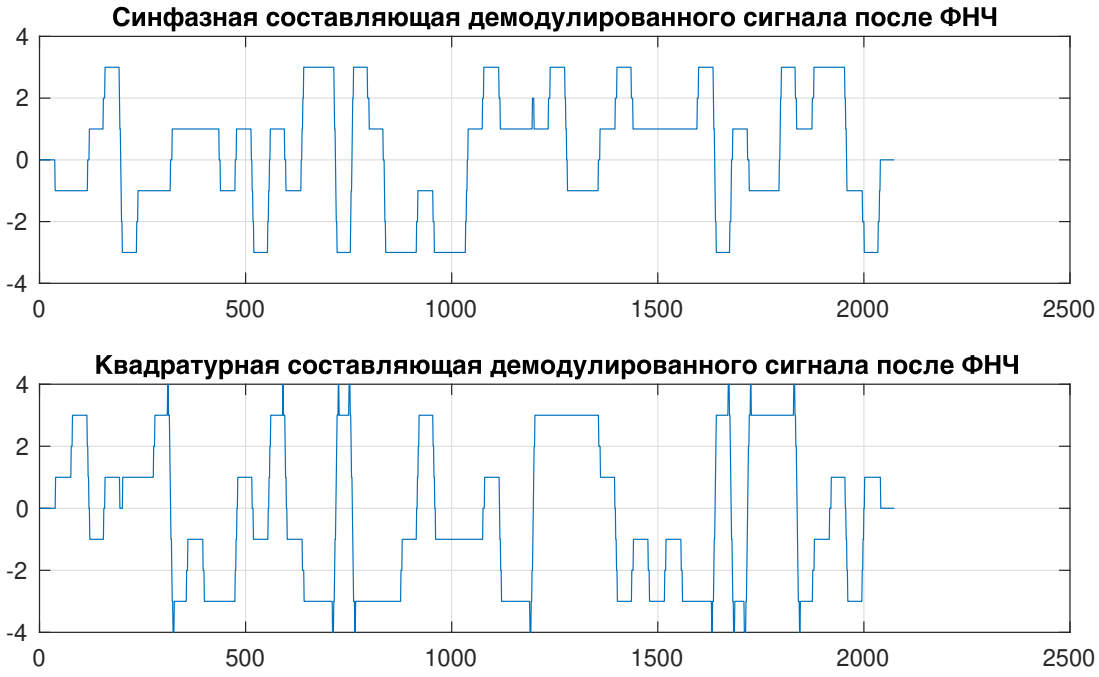

Рис. 12 – Синфазная и квадратурная составляющие демодулированного QASK-сигнала

Следует обратить внимание, что при выделении синфазной и квадратурной составляющей, мы также сделали умножение на два (и получили при этом правильные амплитуды). Давайте разбираться, в чём дело. Как было сказано выше, входной сигнал был умножен на функции $cos(2\pi f_c t)$ и $sin(2\pi f_c t)$. При умножении на косинус получаем сигнал:

$$\begin{aligned} x_{cos}(t) &= (I(t)cos(2\pi f_c t) + Q(t)sin(2\pi f_c t) \quad cos(2\pi f_c t) = \\ &= I(t)cos^2(2\pi f_c t) + \frac{1}{2}Q(t)sin(4\pi f_c t) = \\ &= \frac{1}{2}I(t) + \frac{1}{2}I(t)cos(4\pi f_c t) + \frac{1}{2}Q(t)sin(4\pi f_c t) \end{aligned} \tag{1}$$

При умножении на синус:

$$\begin{aligned} x_{sin}(t) &= (I(t)cos(2\pi f_c t) + Q(t)sin(2\pi f_c t) \quad sin(2\pi f_c t) = \\ &= Q(t)sin^2(2\pi f_c t) + \frac{1}{2}I(t)sin(4\pi f_c t) = \\ &= \frac{1}{2}Q(t) \quad \frac{1}{2}I(t)cos(4\pi f_c t) + \frac{1}{2}Q(t)sin(4\pi f_c t) \end{aligned} \tag{2}$$

После применения ФНЧ косинусоидальные и синусоидальные составляющие уходят, остаётся только постоянная составляющая:

$$x_{cos}(t) = \frac{1}{2}I(t) \tag{3}$$

$$x_{sin}(t) = \frac{1}{2}Q(t) \tag{4}$$

Поэтому, чтобы скомпенсировать амплитуду демодулированного сигнала, в строчках 49 и 50 мы и сделали умножение на 2.

Далее преобразуем `iof` и `qof` в один комплексный сигнал `of`, который проредим, взяв из него отсчёты, расположенные в серединах временных отрезков, соответствующих битам данных. Затем воспользуемся функцией `qamdemod` и сравним результаты: что закодировали, и что в последствии декодировали.

Листинг 12 – QASK-манипуляция, часть 4

```
64 %% Сравнение результатов
65 % формируем комплесный сигнал после фильтра
66 of = complex(iof,qof);
67
68 % считываем из массива значения с шагом,
69 % равным периоду следования данных
70 % и с учётом задержки КИХфильтров-
71 fir_delay = round(length(fltr.Numerator)/2);
72 of_dec = of(fir_delay+round(bit_size/2) : bit_size : length(of)-fir_delay);
73
74 % Сравниваем результаты
75 y = qamdemod(of_dec, M);
76
77 figure
78 plot(data,'b-'), grid on, hold on
79 plot(y,'x','LineWidth',2)
80 title('Сравнение закодированных переданных() и декодированных принятых() данных')
81 xlabel('Номер отсчёта'), ylabel('Значение')
82 legend({'Переданные данные';'Принятые данные'})
```

Синий график на рисунке 13 отображает входные данные, на основе которых был сформирован модулированный сигнал, красными крестиками – данные, полученные в результате демодуляции этого сигнала. Как видим, два графика полностью совпадают, а это говорит о том, что реализованный нами алгоритм работает корректно.

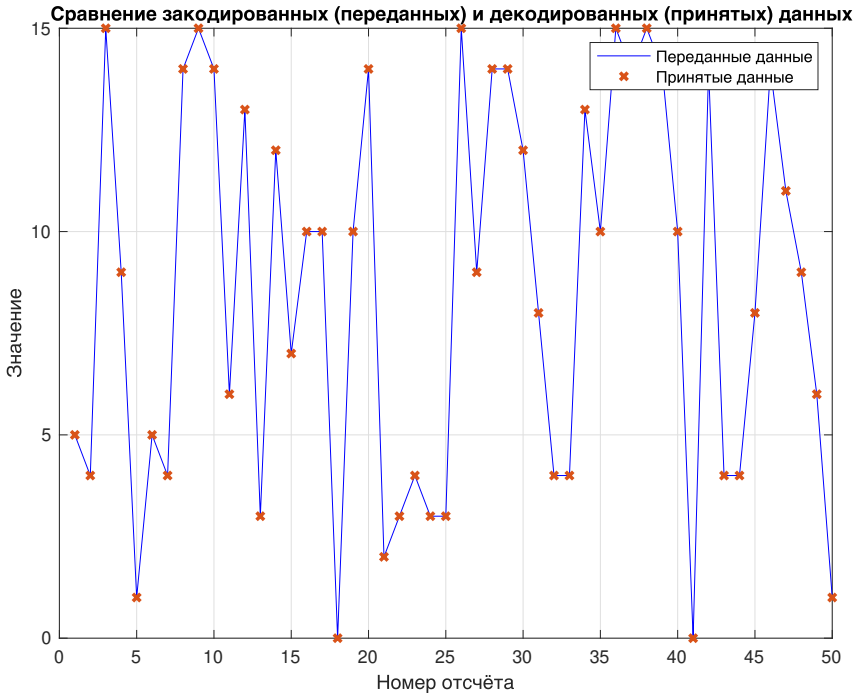

Рис. 13 – Сравнение данных до QASK-модуляции и после демодуляции

5 Частотная манипуляция (FSK)

Если во всех предыдущих рассмотренных нами видах манипуляции частота несущей была постоянна, то в случае с FSK это не так. FSK – это манипуляция, при которой в зависимости от закодированного сообщения скачкообразно изменяется частота несущего сигнала.

Данный вид манипуляции считается самым помехоустойчивым, т.к. помехи чаще всего влияют на амплитуду, а не на несущую частоту. Частота логического «0» и логической «1» вычисляются по формулам:

$$f_0 = f_c - \frac{h}{2T} \tag{5}$$

$$f_1 = f_c + \frac{h}{2T} \tag{6}$$

где:

- f_c – несущая частота
- h – коэффициент модуляции
- T – период передаваемого сообщения

При $h = 0.5$ манипуляция называется **Minimum Shift Keying (MSK)** – манипуляция с минимальным сдвигом частоты.

И снова к примеру. Разработаем скрипт, формирующий FSK-модулированный сигнал. Несущую частоту выберем *500 Гц*, индекс модуляции $h = 4$, период следования битов данных 10 мкс. На основе этого рассчитаем значение двух частот: `f0` и `f1`, кодирующие логический «0» и «1» соответственно.

Листинг 13 – FSK-манипуляция, часть 1

```
clear, clc, close all
fs = 10000;
ts = 0 : 1/fs : 0.1-1/fs;
N = length(ts);

%% параметры несущей
fc = 500;           % несущая частота
h = 4;              % индекс модуляции
T = 10e-3;          % длительность бита

f0 = fc+h/(2*T);    % частота лог. "0"
f1 = fc-h/(2*T);    % частота лог. "1"
```

Далее сформируем модулирующий сигнал по аналогии с тем, как мы это делали в примерах с BPSK и ASK:

Листинг 14 – FSK-манипуляция, часть 2

```
%% модулирующий сигнал
% длина одного бита в отсчётах
n_for_bit = T*fs;
% кодируемая последовательность
code = [1 0 0 1 1 0 1 0 1 1];

% формируем модулирующий сигнал
fm = zeros(1,N);
for i=1:length(code)
    for j=n_for_bit*(i-1)+1:n_for_bit*i
        fm(j) = code(i);
    end
end
```

Теперь создадим два сигнала: `x0` с частотой `f0` и `x1` с частотой `f1`. На базе этих сигналов и модулирующего сигнала `fm` сформируем модулированный сигнал `x` и построим графики.

Листинг 15 – FSK-манипуляция, часть 3

```
%% Частотная манипуляция
x0 = cos(2*pi*f0*ts);
x1 = cos(2*pi*f1*ts);

```

```
x = zeros(1,N);
for i = 1:N
    if fm(i) == 0
        x(i) = x0(i);
    else
        x(i) = x1(i);
    end
end

plot(ts,x,'LineWidth',0.5), grid on, hold on
plot(ts,fm,'LineWidth',2), grid on
title ('FSKмодуляция-')
xlabel('Время'), ylabel('Амплитуда')
legend({'Модулированный сигнал';'Модулирующий сигнал'})
```

Результат выполнения данного скрипта показан на рисунке 14. Из рисунка видно, что в зависимости от зна-

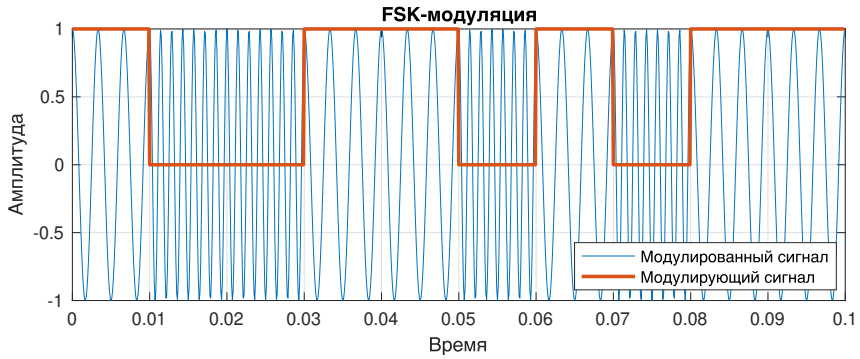

Рис. 14 – FSK-манипуляция

чения модулирующего сигнала, меняется частота несущей – это то, что нам нужно. Модулировать научились – теперь попробуем это демодулировать. Для этого умножим наш сигнал на косинусоиды с частотами `f0` и `f1`:

Листинг 16 – FSK-манипуляция, часть 4

```
%% FSKдемодуляция-
% умножаем сигнал на несущую
y0 = x.*x0;
y1 = x.*x1;

figure;
subplot(2,1,1)
plot(ts,y0), grid on
```

```
52 title('Модулированный сигнал, умноженный на f0')
53 xlabel('Время'), ylabel('Амплитуда')
54 subplot(2,1,2)
55 plot(ts,y1), grid on
56 title('Модулированный сигнал, умноженный на f1')
57 xlabel('Время'), ylabel('Амплитуда')
```

Сигналы `y0` и `y1` показаны на рисунке 13. Теперь найдём разность этих сигналов и пропустим её через ФНЧ:

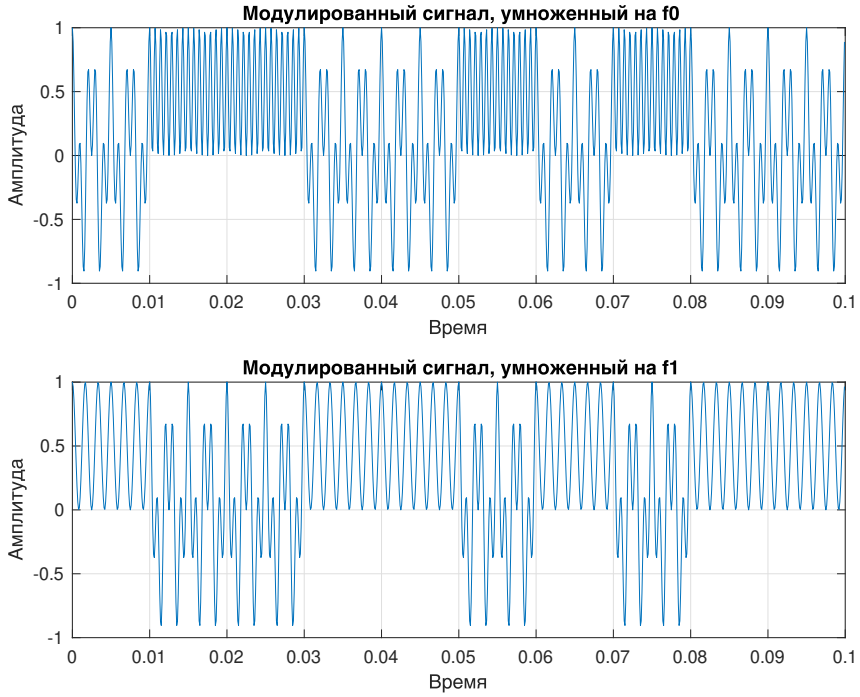

Рис. 15 – Результаты выполнения листинга 16

Листинг 17 – FSK-манипуляция, часть 5

```
58 y = y1 - y0;
59
60 % применяем ФНЧ, чтобы оставить постоянную составляющую
61 fltr = fsk_fir;
62 z = filter(fltr.Numerator,1,y);
63
64 figure
65 subplot(2,1,1)
66 plot(ts,y), grid on
67 title('Сигнал y=y0-y1')
```

```
xlabel('Время'), ylabel('Амплитуда')
subplot(2,1,2)
plot(ts,z), grid on
title('Этот же сигнал после ФНЧ')
xlabel('Время'), ylabel('Амплитуда')
```

Результат показан на рисунке 15. До финала осталось совсем немного – преобразовать нижний график в

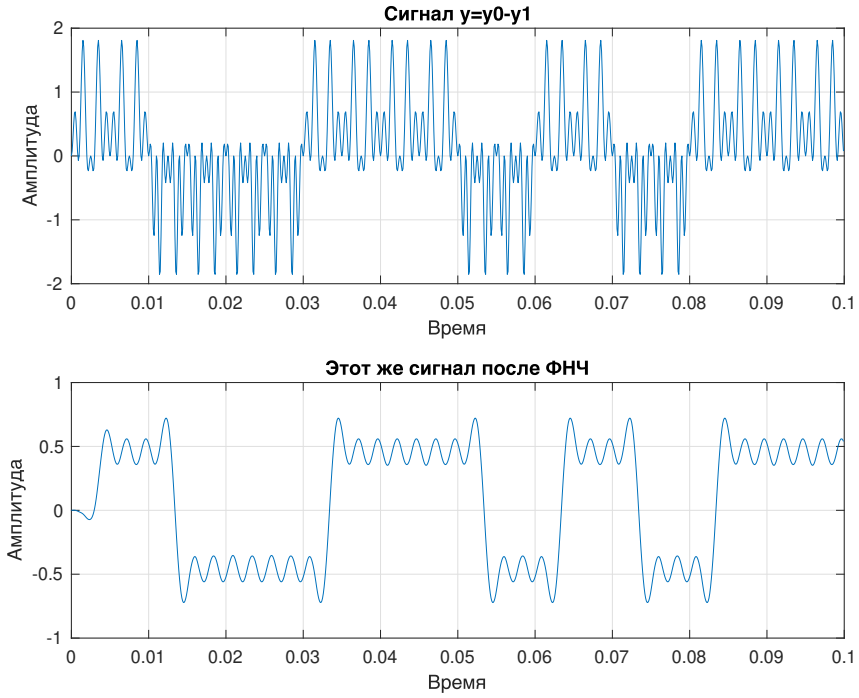

Рис. 16 – Результаты выполнения листинга 17

цифровой сигнал. Для этого, как и в случае с BPSK, воспользуемся цифровым компаратором и построим результирующий график:

Листинг 18 – FSK-манипуляция, часть 6

```
% компаратор
dem = zeros(1,length(z));
for i=1:length(z)
    if z(i)>=0.1
        dem(i) = 1;
    else
        dem(i) = 0;
    end
end
```

```
82
83 figure
84 plot(ts,x,'LineWidth',0.5), grid on, hold on
85 plot(ts,dem,'LineWidth',2), grid on
86 xlabel('Время'), ylabel('Амплитуда')
87 title ('FSKдемодуляция-')
88 legend({'Модулированный сигнал';'Демодулированный сигнал'})
```

Результат демодуляции показан на рисунке 17. Как видим, оранжевый график имеет ту же форму, что и на

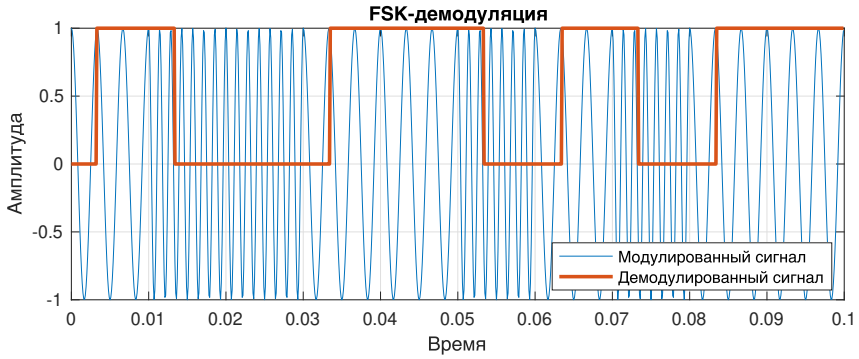

Рис. 17 – FSK-демодуляция

рисунке 14, но с уже привычной нам временной задержкой, возникающей в цифровом КИХ-фильтре.

6 Задания на семинар

1. Разработать скрипт, формирующий аналитический сигнал из произвольного действительного, не используя функцию `hilbert`.

2. Разработать скрипты и модели Simulink, которые для произвольного сигнала реализуют модуляции и демодуляции вида: ASK, 8-PSK, 64-QAM, MSK. Для каждого вида модуляции построить сигнальное созвездие.