

# **Лабораторная работа № 1.**

**Методы кодирования и модуляция сигналов**

Диана Алексеевна Садова

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>6</b>
<b>2</b>	<b>Последовательность выполнения работы</b>	<b>7</b>
2.1	Построение графиков в Octave . . . . .	7
2.1.1	Постановка задачи . . . . .	7
2.1.2	Порядок выполнения работы . . . . .	7
2.2	Разложение импульсного сигнала в частичный ряд Фурье . . . . .	14
2.2.1	Постановка задачи . . . . .	14
2.2.2	Порядок выполнения работы . . . . .	15
2.3	Определение спектра и параметров сигнала . . . . .	19
2.3.1	Постановка задачи . . . . .	19
2.3.2	Порядок выполнения работы . . . . .	20
2.4	Амплитудная модуляция . . . . .	29
2.4.1	Постановка задачи . . . . .	29
2.4.2	Порядок выполнения работы . . . . .	29
2.5	Кодирование сигнала. Исследование свойства самосинхронизации сигнала . . . . .	32
2.5.1	Постановка задачи . . . . .	32
2.5.2	Порядок выполнения работы . . . . .	32
<b>3</b>	<b>Выводы</b>	<b>59</b>
	<b>Список литературы</b>	<b>60</b>

## Список иллюстраций

2.1	Загружаем Octave . . . . .	8
2.2	Процес установки . . . . .	9
2.3	Octave . . . . .	10
2.4	ОС Octave с оконным интерфейсом . . . . .	11
2.5	Код функции $y = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$ . . . . .	12
2.6	График функции $y = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$ . . . . .	13
2.7	График функций $y_1$ и $y_2$ . . . . .	14
2.8	Код meandr.m . . . . .	15
2.9	Код meandr.m . . . . .	16
2.10	Код meandr.m . . . . .	17
2.11	График cos . . . . .	18
2.12	График sin . . . . .	19
2.13	Код spectre.m . . . . .	20
2.14	Два синусоидальных сигнала разной частоты . . . . .	21
2.15	Дополнение в коду spectre.m . . . . .	22
2.16	Дополнение в коду spectre.m . . . . .	23
2.17	График спектров синусоидальных сигналов . . . . .	24
2.18	Исправленный график спектров синусоидальных сигналов . . . . .	25
2.19	Код spectre_sum.m . . . . .	26
2.20	Суммарный сигнал . . . . .	27
2.21	Спектр суммарного сигнала . . . . .	28
2.22	Код am.m . . . . .	30
2.23	Сигнал и огибающая при амплитудной модуляции . . . . .	31
2.24	Спектр сигнала при амплитудной модуляции . . . . .	32
2.25	Окно терминала. Проверяем наличие signal . . . . .	33
2.26	Код main.m . . . . .	33
2.27	Код main.m . . . . .	34
2.28	Код main.m . . . . .	35
2.29	Код main.m . . . . .	36
2.30	Код maptowave.m . . . . .	37
2.31	Код unipolar.m . . . . .	37
2.32	Код ami.m . . . . .	38
2.33	Код bipolarnrz.m . . . . .	38
2.34	Код bipolarrrz.m . . . . .	38
2.35	Код manchester.m . . . . .	39
2.36	Код diffmanc.m . . . . .	39
2.37	Код calcspectre.m . . . . .	40

2.38	Униполярное кодирование . . . . .	41
2.39	Кодирование AMI . . . . .	42
2.40	Кодирование NRZ . . . . .	43
2.41	Кодирование RZ . . . . .	44
2.42	Манчестерское кодирование . . . . .	45
2.43	Дифференциальное манчестерское кодирование . . . . .	46
2.44	Униполярное кодирование: нет самосинхронизации . . . . .	47
2.45	Кодирование AMI: самосинхронизация при наличии сигнала . . .	48
2.46	Кодирование NRZ: нет самосинхронизации . . . . .	49
2.47	Кодирование RZ: есть самосинхронизация . . . . .	50
2.48	Манчестерское кодирование: есть самосинхронизация . . . . .	51
2.49	Дифференциальное манчестерское кодирование: есть самосин- хронизация . . . . .	52
2.50	Униполярное кодирование: спектр сигнала . . . . .	53
2.51	Кодирование AMI: спектр сигнала . . . . .	54
2.52	Кодирование NRZ: спектр сигнала . . . . .	55
2.53	Кодирование RZ: спектр сигнала . . . . .	56
2.54	Манчестерское кодирование: спектр сигнала . . . . .	57
2.55	Дифференциальное манчестерское кодирование: спектр сигнала	58

## **Список таблиц**

# 1 Цель работы

Изучение методов кодирования и модуляции сигналов с помощью высокоуровневого языка программирования Octave. Определение спектра и параметров сигнала. Демонстрация принципов модуляции сигнала на примере аналоговой амплитудной модуляции. Исследование свойства самосинхронизации сигнала.

## 2 Последовательность выполнения работы

### 2.1 Построение графиков в Octave

#### 2.1.1 Постановка задачи

1. Построить график функции  $y = \sin x + \frac{1}{3} \sin 3x + \frac{1}{5} \sin 5x$  на интервале  $[-10; 10]$ , используя Octave и функцию `plot`. График экспортировать в файлы формата `.eps`, `.png`.
2. Добавить график функции  $y = \cos x + \frac{1}{3} \cos 3x + \frac{1}{5} \cos 5x$  на интервале  $[-10; 10]$ . График экспортировать в файлы формата `.eps`, `.png`.

#### 2.1.2 Порядок выполнения работы

1. Запустите в вашей ОС Octave с оконным интерфейсом.(рис. 2.1),(рис. 2.2),(рис. 2.3),(рис. 2.4).

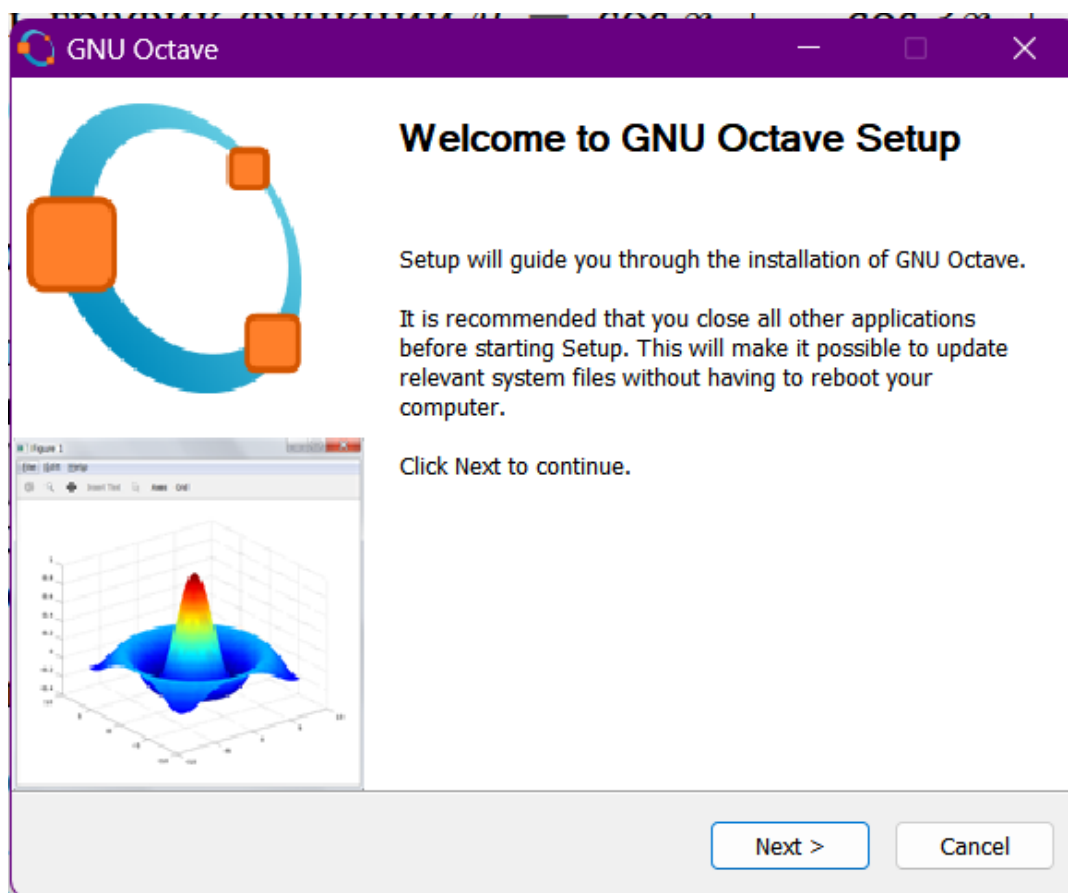


Рис. 2.1: Загружаем Octave



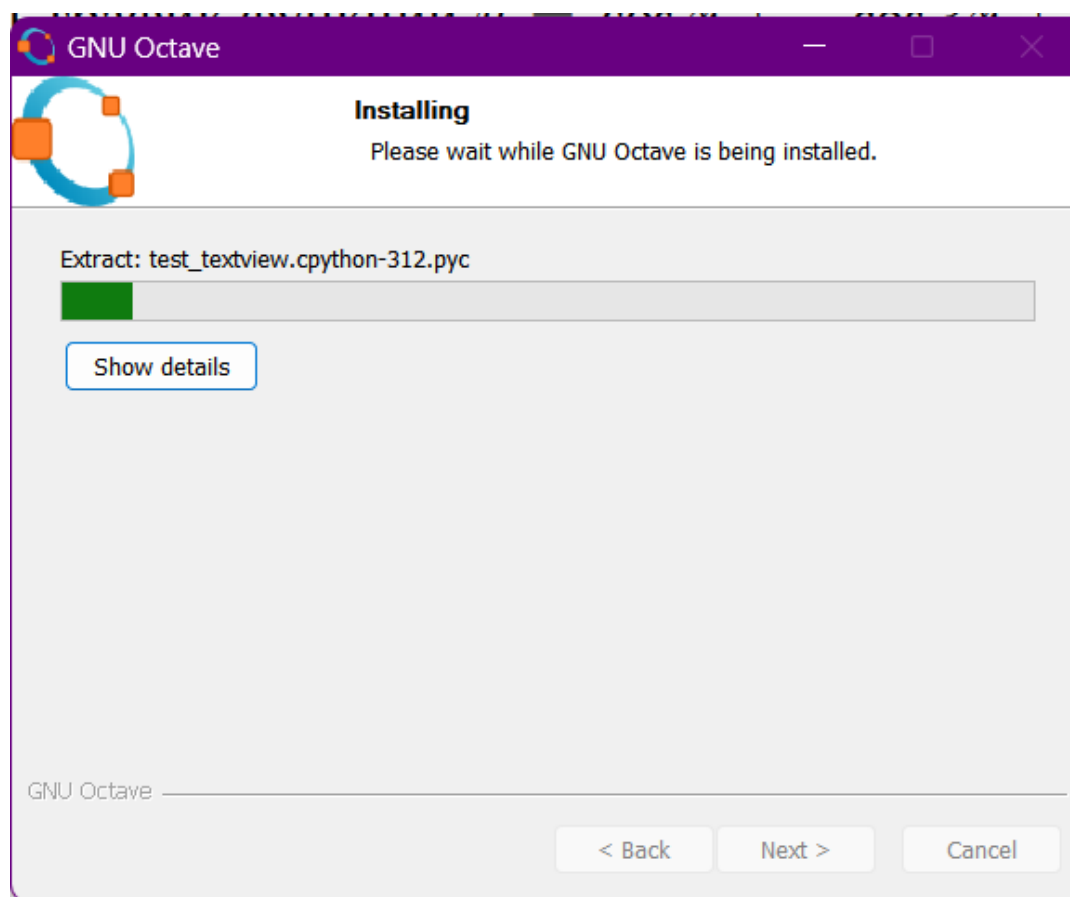


Рис. 2.2: Процесс установки

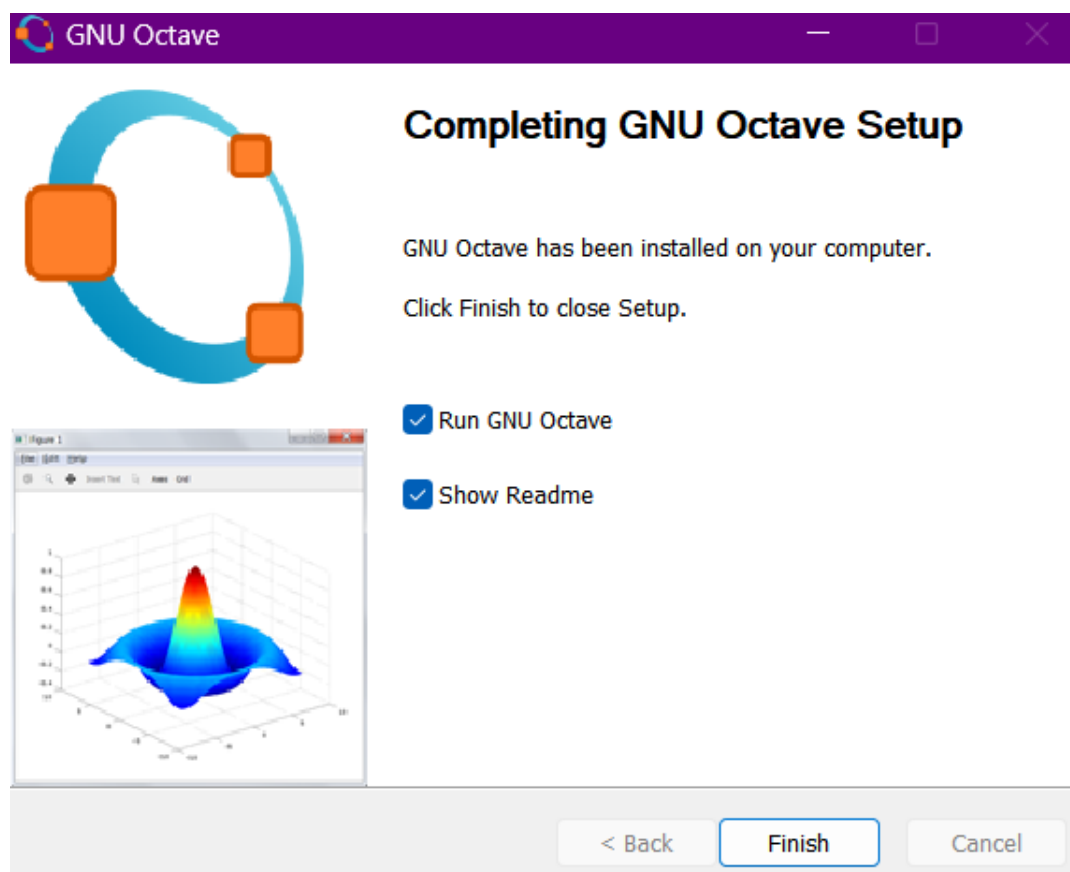


Рис. 2.3: Octave

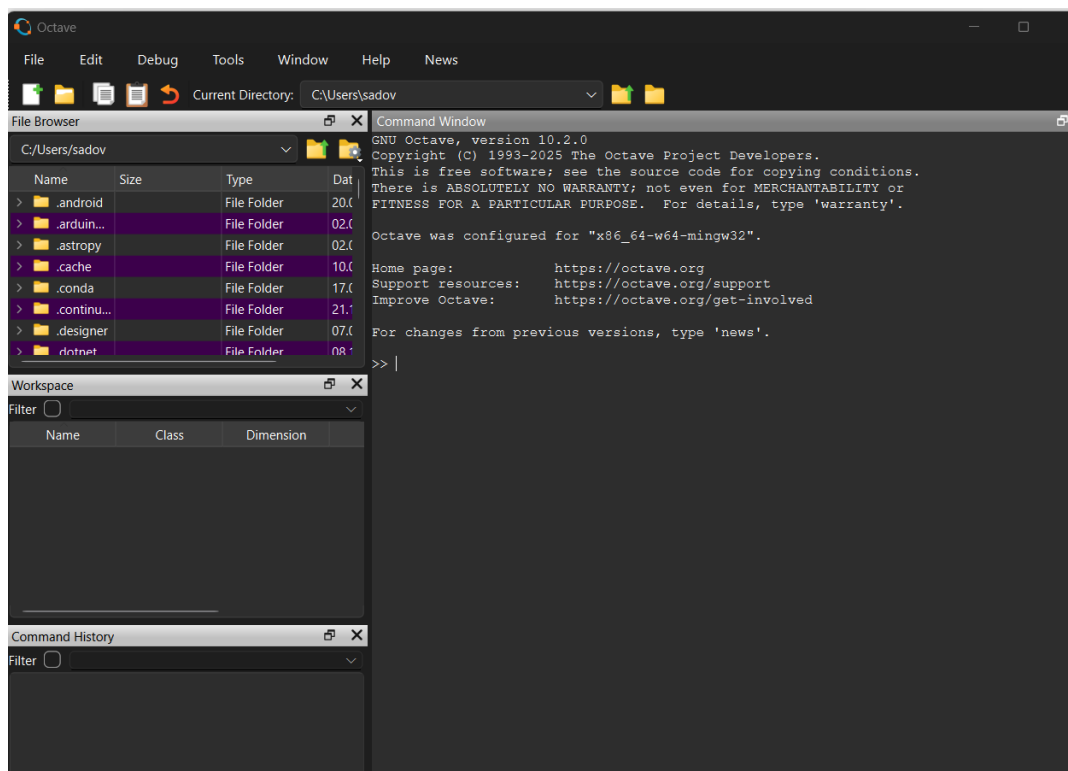


Рис. 2.4: ОС Octave с оконным интерфейсом

2. Перейдите в окно редактора. Воспользовавшись меню или комбинацией клавиш `ctrl + n` создайте новый сценарий. Сохраните его в ваш рабочий каталог с именем, например, `plot_sin.m`.
3. В окне редактора повторите следующий листинг по построению графика функции  $y = \sin x + \frac{1}{3} \sin 3x + \frac{1}{5} \sin 5x$  на интервале  $[-10; 10]$ : (рис. 2.5).

```

1 % Формирование массива x:
2 x=-10:0.1:10;
3 % Формирование массива y.
4 y1=sin(x)+1/3*sin(3*x)+1/5*sin(5*x);
5 % Построение графика функции:
6 plot(x,y1, "-ok; y1=sin(x)+
7   (1/3)*sin(3*x)+(1/5)*sin(5*x);", "markersize", 4)
8 % Отображение сетки на графике
9 grid on;
10 % Подпись оси X:
11 xlabel('x');
12 % Подпись оси Y:
13 ylabel('y');
14 % Название графика:
15 title('y1=sin x+ (1/3) sin(3x)+(1/5) sin(5x)');
16 % Экспорт рисунка в файл .eps:
17 print ("plot-sin.eps", "-mono", "-FArial:16", "-deps")
18 % Экспорт рисунка в файл .png:
19 print("plot-sin.png");
20

```

Рис. 2.5: Код функции  $y = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$

В нашем случае имя test.m

4. Запустите сценарий на выполнение (воспользуйтесь соответствующим меню окна редактора или клавишей F5 ). В качестве результата выполнения кода должно открыться окно с построенным графиком (рис. 1.1) и в вашем рабочем каталоге должны появиться файлы с графиками в форматах .eps, .png. (рис. 2.6).

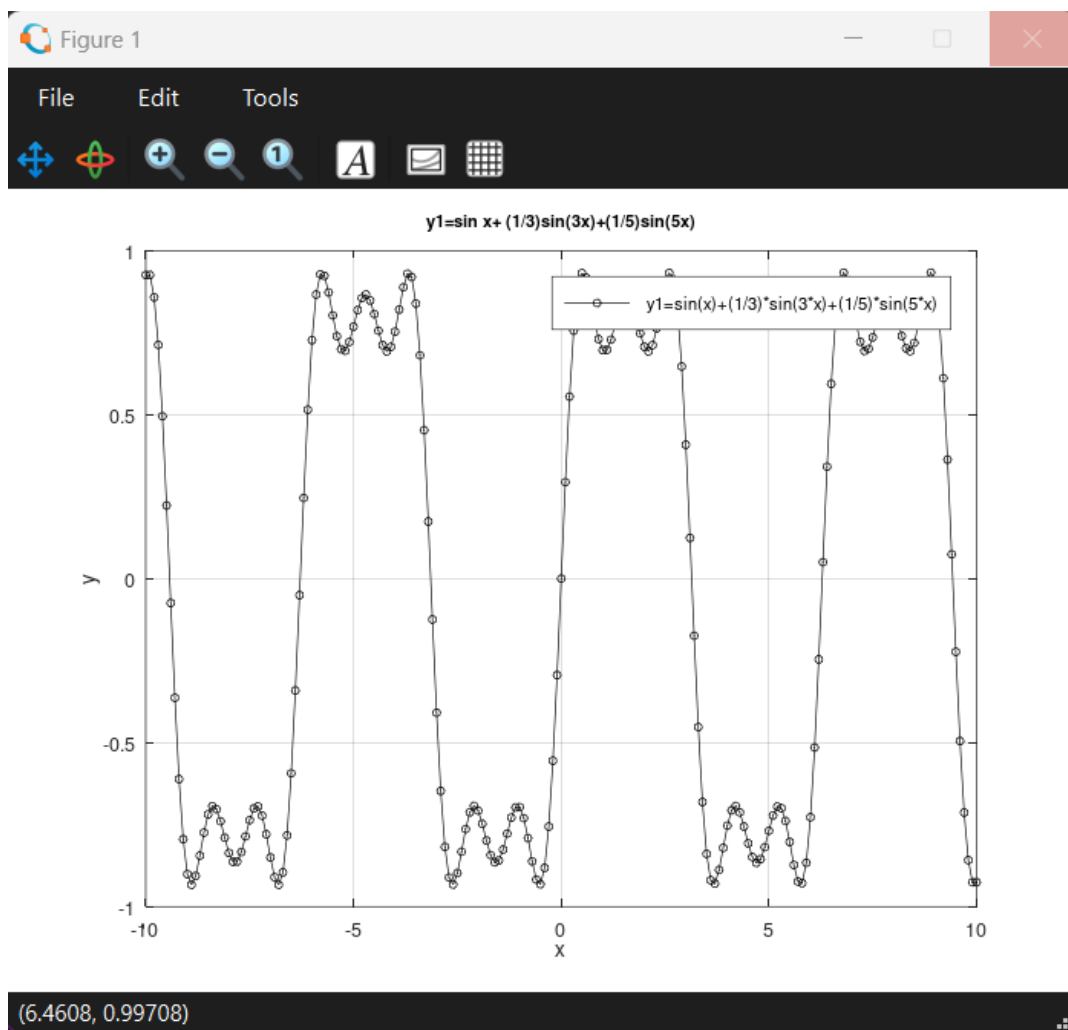


Рис. 2.6: График функции  $y = \sin x + \frac{1}{3} \sin 3x + \frac{1}{5} \sin 5x$

5. Сохраните сценарий под другим названием и измените его так, чтобы на одном графике располагались отличающиеся по типу линий графики функций  $y_1 = \sin x + \frac{1}{3} \sin 3x + \frac{1}{5} \sin 5x$ ,  $y_2 = \cos x + \frac{1}{3} \cos 3x + \frac{1}{5} \cos 5x$ , например как изображено (рис. 2.7).

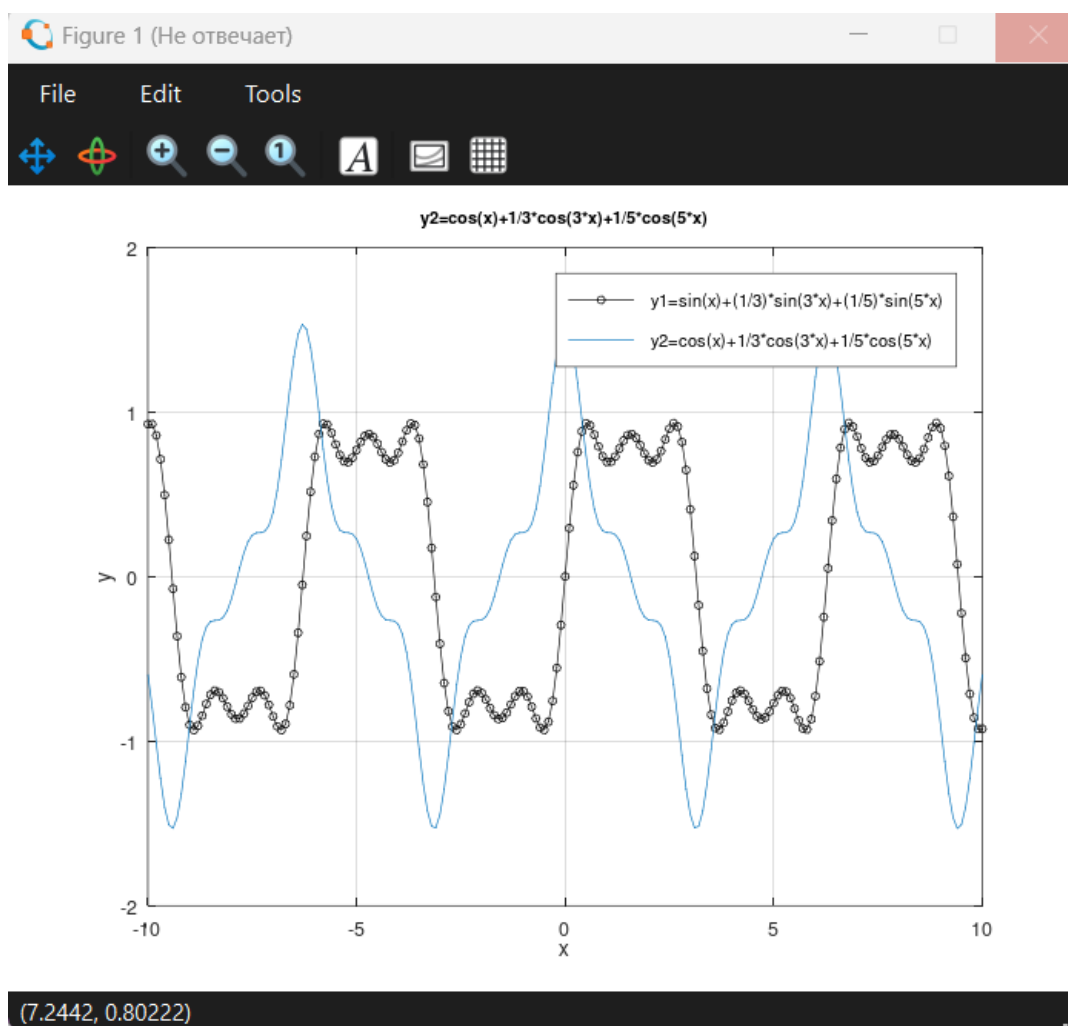


Рис. 2.7: График функций  $y_1$  и  $y_2$

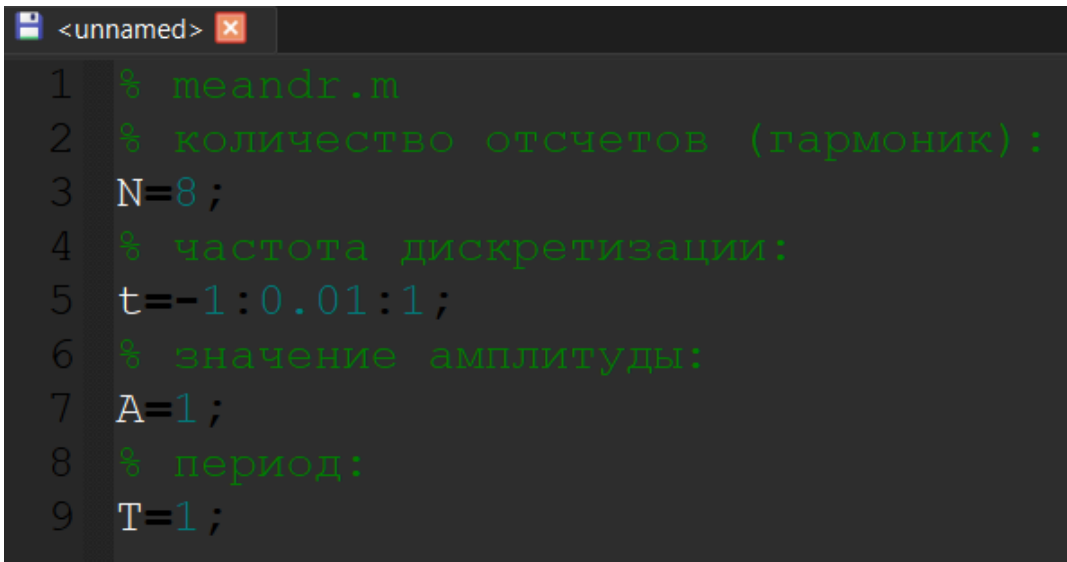
## 2.2 Разложение импульсного сигнала в частичный ряд Фурье

### 2.2.1 Постановка задачи

1. Разработать код m-файла, результатом выполнения которого являются графики меандра, реализованные с различным количеством гармоник.

### 2.2.2 Порядок выполнения работы

1. Создайте новый сценарий и сохраните его в ваш рабочий каталог с именем, например, meandr.m.
2. В коде созданного сценария задайте начальные значения:(рис. 2.8).



```
<unnamed>
1 % meandr.m
2 % количество отсчетов (гармоник):
3 N=8;
4 % частота дискретизации:
5 t=-1:0.01:1;
6 % значение амплитуды:
7 A=1;
8 % период:
9 T=1;
```

Рис. 2.8: Код meandr.m

3. Разложение импульсного сигнала в форме меандра в частичный ряд Фурье. Гармоники, образующие меандр, имеют амплитуду, обратно пропорциональную номеру соответствующей гармоники в спектре:(рис. 2.9).

```

1 % meandr.m
2 % количество отсчетов (гармоник):
3 N=8;
4 % частота дискретизации:
5 t=-1:0.01:1;
6 % значение амплитуды:
7 A=1;
8 % период:
9 T=1;
10 % амплитуда гармоник
11 nh=(1:N)*2-1;
12 % массив коэффициентов для ряда, заданного через cos:
13 Am=2/pi ./ nh;
14 Am(2:2:end) = -Am(2:2:end);
15
16 % массив гармоник:
17 harmonics=cos(2 * pi * nh' * t/T);
18 % массив элементов ряда:
19 s1=harmonics.*repmat(Am',1,length(t));
20

```

Рис. 2.9: Код meandr.m

4. Далее для построения в одном окне отдельных графиков меандра с различным количеством гармоник реализуем суммирование ряда с накоплением и воспользуемся функциями subplot и plot для построения графиков:(рис. 2.10).



```

1 % meandr.m
2 % количество отсчетов (гармоник):
3 N=8;
4 % частота дискретизации:
5 t=-1:0.01:1;
6 % значение амплитуды:
7 A=1;
8 % период:
9 T=1;
10 % амплитуда гармоник
11 nh=(1:N)*2-1;
12 % массив коэффициентов для ряда, заданного через cos:
13 Am=2/pi ./ nh;
14 Am(2:2:end) = -Am(2:2:end);
15
16 % массив гармоник:
17 harmonics=cos(2 * pi * nh' * t/T);
18 % массив элементов ряда:
19 s1=harmonics.*repmat(Am',1,length(t));
20
21 % Суммирование ряда:
22 s2=cumsum(s1);
23 % Построение графиков:
24 for k=1:N
25     subplot(4,2,k)
26     plot(t, s2(k,:))
27 end

```

Рис. 2.10: Код meandr.m

5. Экпортируйте полученный график в файл в формате .png.(рис. 2.11).

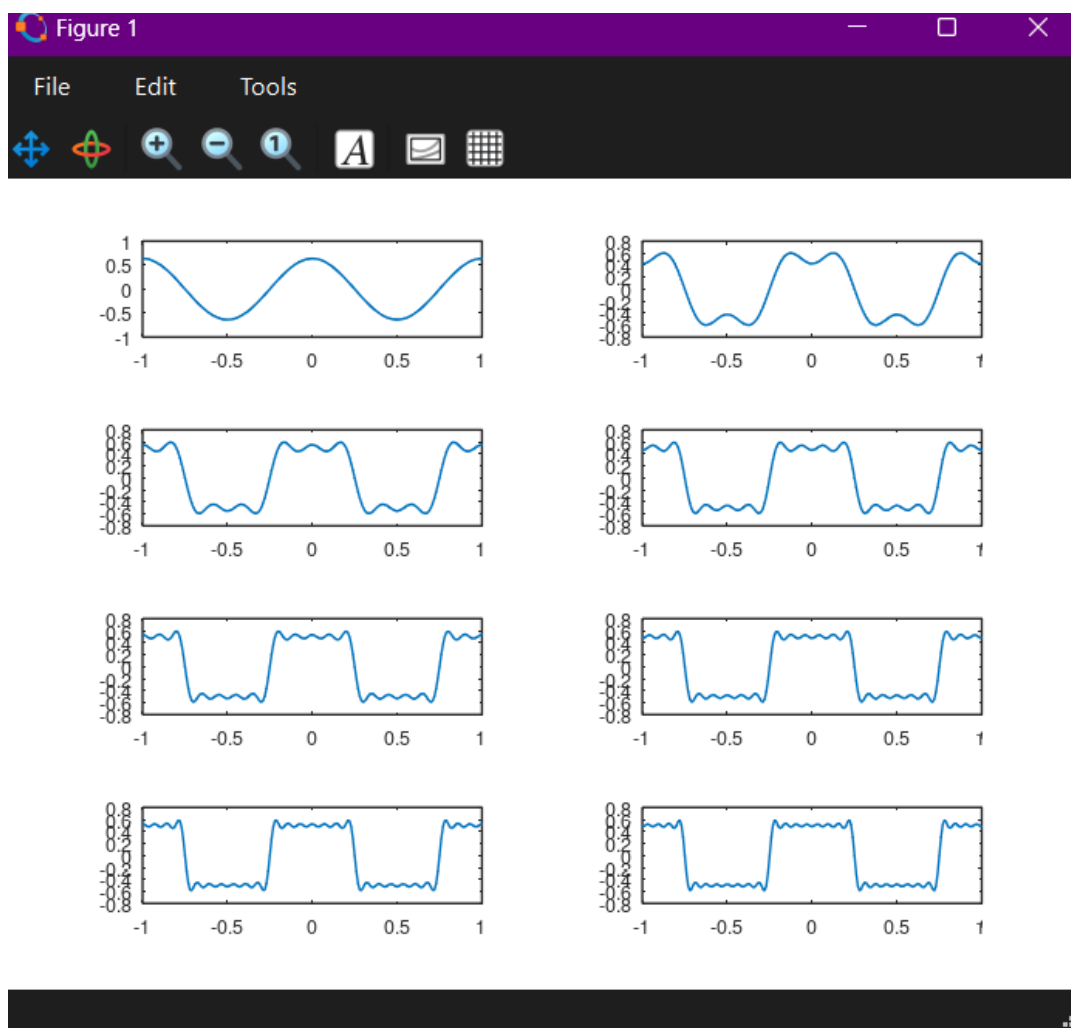


Рис. 2.11: График  $\cos$

6. Скорректируйте код для реализации меандра через синусы. Получите соответствующие графики.(рис. 2.12).

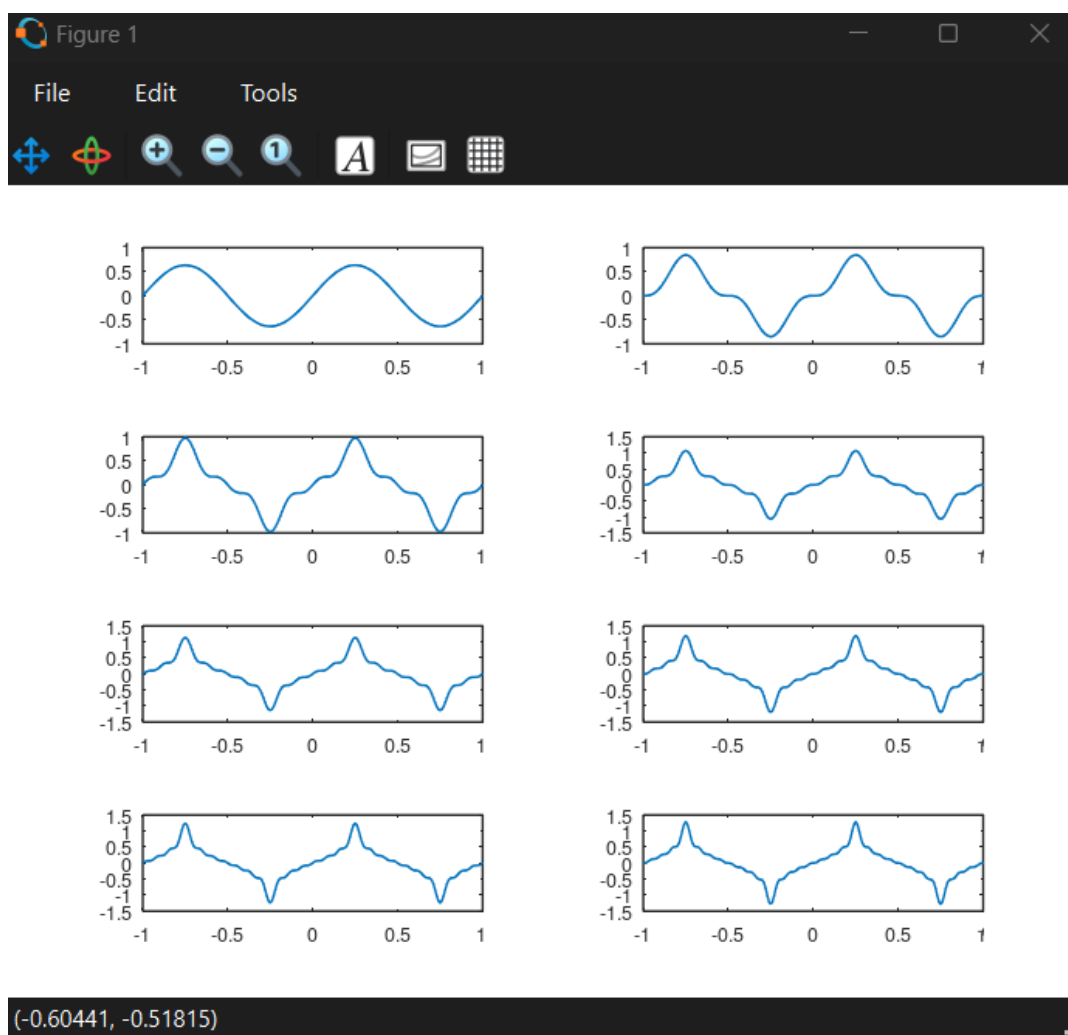


Рис. 2.12: График  $\sin$

## 2.3 Определение спектра и параметров сигнала

### 2.3.1 Постановка задачи

1. Определить спектр двух отдельных сигналов и их суммы.
2. Выполнить задание с другой частотой дискретизации. Пояснить, что будет, если взять частоту дискретизации меньше 80 Гц?

### 2.3.2 Порядок выполнения работы

1. В вашем рабочем каталоге создайте каталог spectre1 и в нём новый сценарий с именем, spectre.m.
2. В коде созданного сценария задайте начальные значения:
3. Далее в коде задайте два синусоидальных сигнала разной частоты:
4. Постройте графики сигналов:(рис. 2.13),(рис. 2.14).

```
% spectre1/spectre.m
% Создание каталогов signal и spectre для размещения графиков:
mkdir 'signal';
mkdir 'spectre';
% Длина сигнала (с):
tmax = 0.5;
% Частота дискретизации (Гц) (количество отсчетов):
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;
% Массив отсчетов времени:
t = 0:1./fd:tmax;
% Спектр сигнала:
fd2 = fd/2;

% Два сигнала разной частоты:
signal1 = a1*sin(2*pi*t*f1);
signal2 = a2*sin(2*pi*t*f2);

% График 1-го сигнала:
plot(signal1,'b');
% График 2-го сигнала:
hold on
plot(signal2,'r');
hold off
title('Signal');
% Экспорт графика в файл в каталоге signal:
print 'signal/spectre.png';
```

Рис. 2.13: Код spectre.m

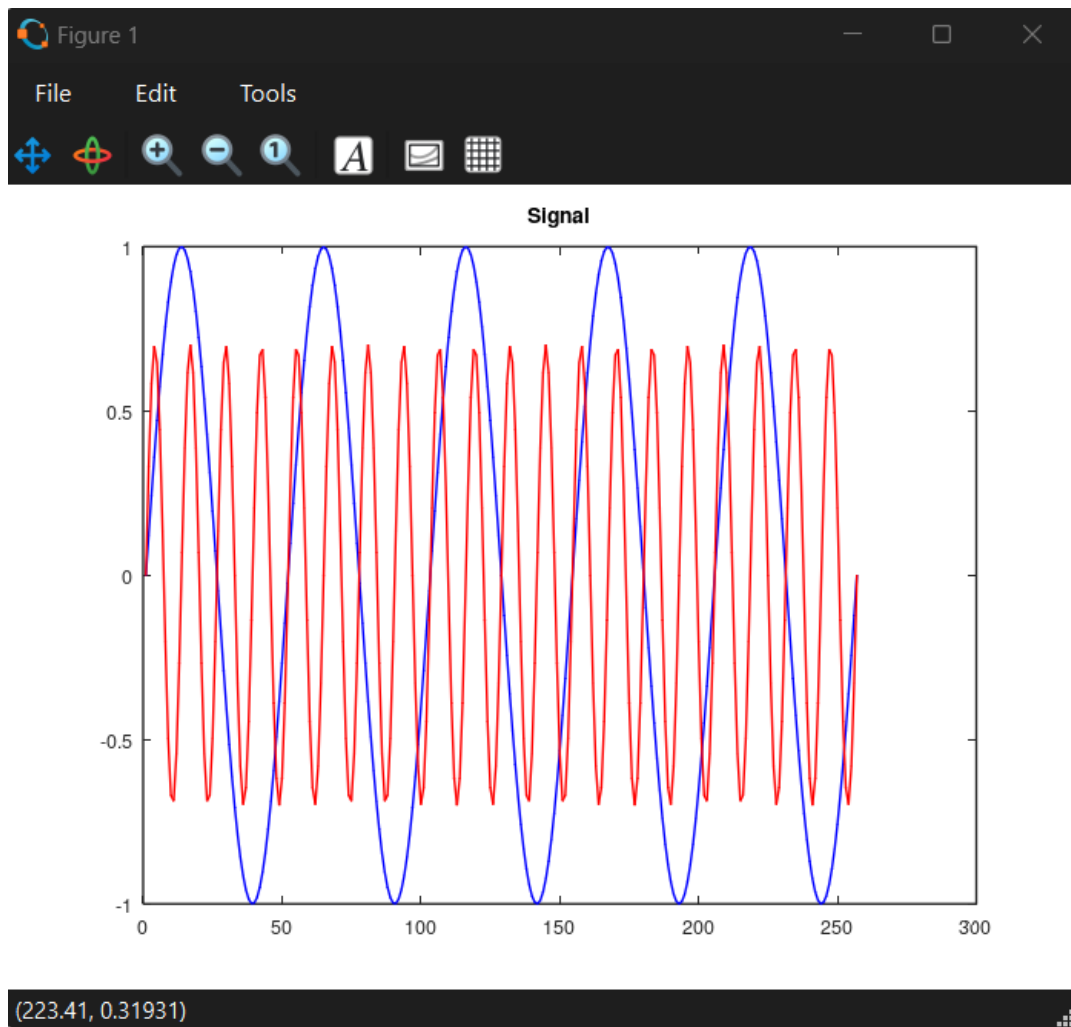


Рис. 2.14: Два синусоидальных сигнала разной частоты

5. С помощью быстрого преобразования Фурье найдите спектры сигналов, добавив в файл `sprectre.m` следующий код:(рис. 2.15).

```

% Посчитаем спектр
% Амплитуды преобразования Фурье сигнала 1:
spectre1 = abs(fft(signal1,fd));
% Амплитуды преобразования Фурье сигнала 2:
spectre2 = abs(fft(signal2,fd));
% Построение графиков спектров сигналов:
plot(spectre1,'b');
hold on
plot(spectre2,'r');
hold off
title('Spectre');
print 'spectre/spectre.png';

```

Рис. 2.15: Дополнение в коду spectre.m

6. Учитывая реализацию преобразования Фурье, скорректируйте график спектра: отбросьте дублирующие отрицательные частоты, а также примите в расчёт то, что на каждом шаге вычисления быстрого преобразования Фурье происходит суммирование амплитуд сигналов. Для этого добавьте в файл spectre.m следующий код:(рис. 2.16),(рис. 2.17),(рис. 2.18).

```

% Посчитаем спектр
% Амплитуды преобразования Фурье сигнала 1:
spectre1 = abs(fft(signal1,fd));
% Амплитуды преобразования Фурье сигнала 2:
spectre2 = abs(fft(signal2,fd));
% Построение графиков спектров сигналов:
plot(spectre1,'b');
hold on
plot(spectre2,'r');
hold off
title('Spectre');
print 'spectre/spectre.png';

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

plot(f,spectre2(1:fd2+1),'r');
hold off
xlim([0 100]);
title('Fixed spectre');
xlabel('Frequency (Hz)');
print 'spectre/spectre_fix.png';

```

Рис. 2.16: Дополнение в коду spectre.m

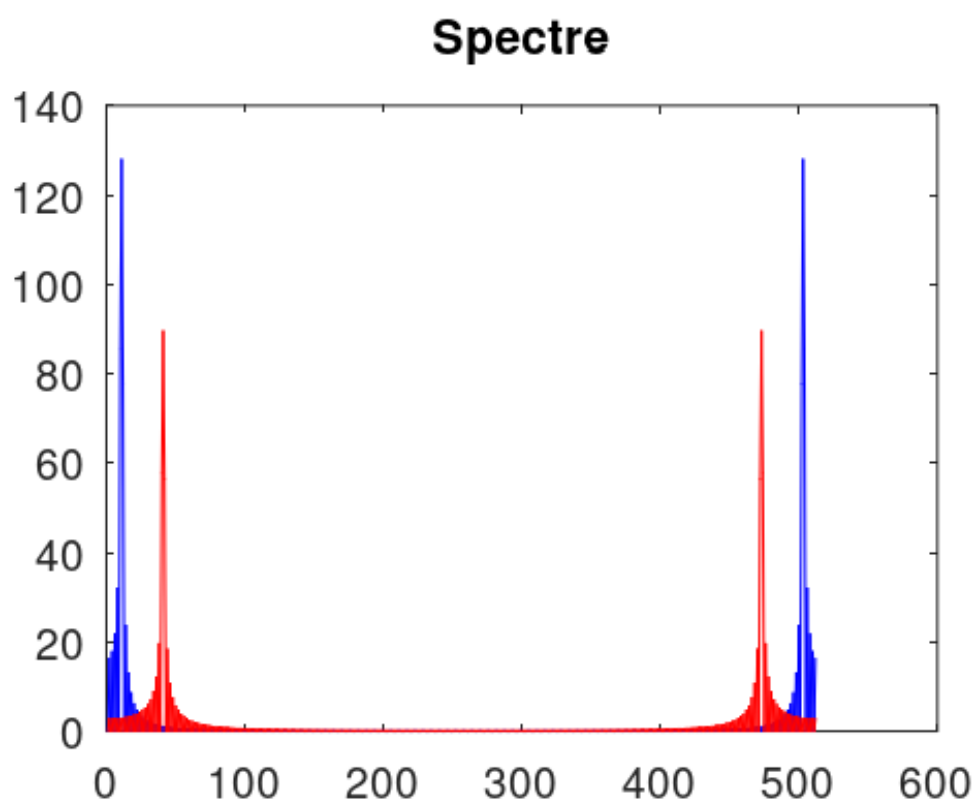


Рис. 2.17: График спектров синусоидальных сигналов



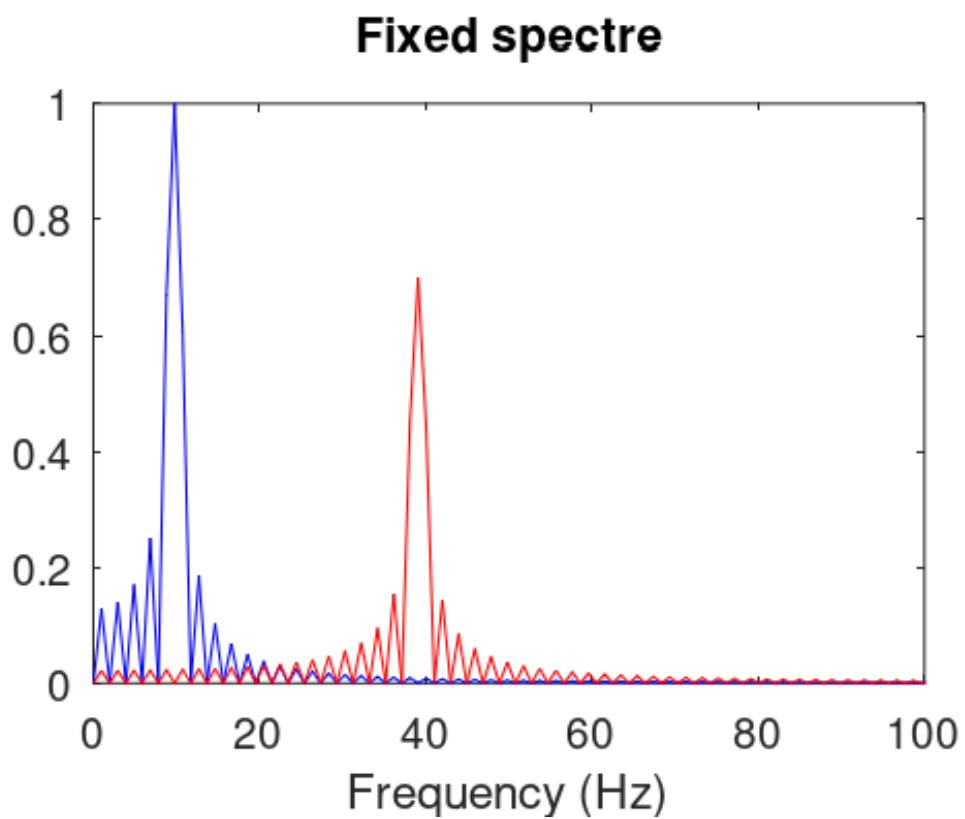


Рис. 2.18: Исправленный график спектров синусоидальных сигналов

7. Найдите спектр суммы рассмотренных сигналов, создав каталог `spectr_sum` и файл в нём `spectre_sum.m` со следующим кодом:(рис. 2.19).

```

% spectr_sum/spectre_sum.m
% Создание каталогов signal и spectre для размещения графиков:
mkdir 'signal';
mkdir 'spectre';
% Длнна сигнала (с):
tmax = 0.5;
% Частота дискретизации (Гц) (количество отсчетов):
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 отсчетов времени:
t = 0:1./fd:tmax;
signal1 = a1*sin(2*pi*t*f1);

signal2 = a2*sin(2*pi*t*f2);
signal = signal1 + signal2;
plot(signal);
title('Signal');
print 'signal/spectre_sum.png';
% 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 графика спектра сигнала:
plot(f,spectre(1:fd2+1))
xlim([0 100]);
title('Spectre');
xlabel('Frequency (Hz)');
print 'spectre/spectre_sum.png';

```

Рис. 2.19: Код spectre\_sum.m

В результате должен получится аналогичный предыдущему результат, т.е. спектр суммы сигналов должен быть равен сумме спектров сигналов, что вытекает из свойств преобразования Фурье.(рис. 2.20),(рис. 2.21).

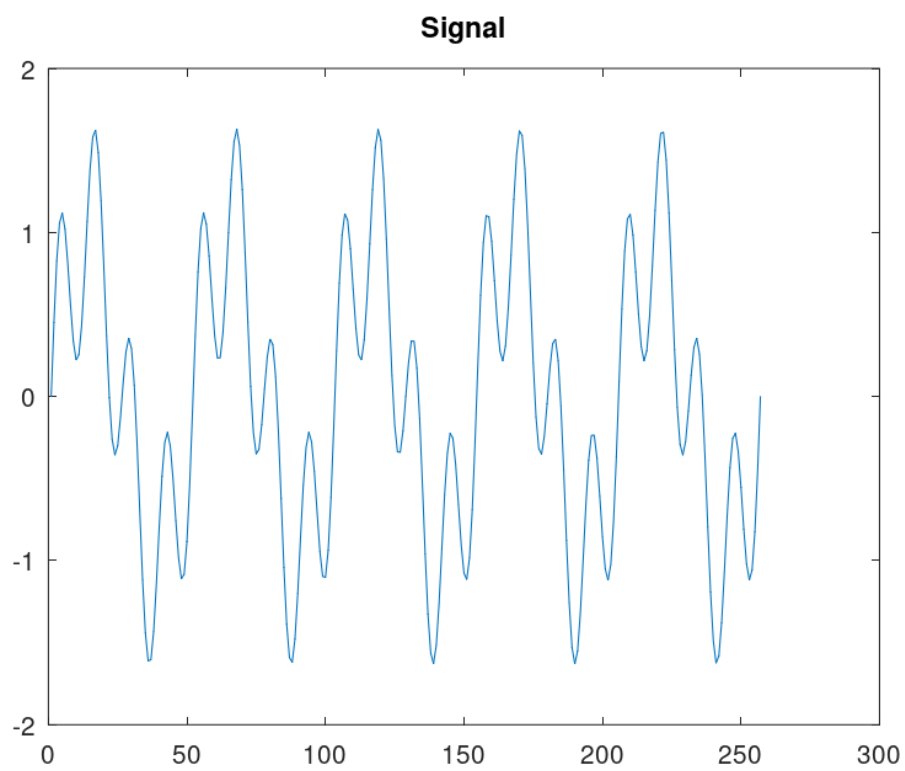


Рис. 2.20: Суммарный сигнал

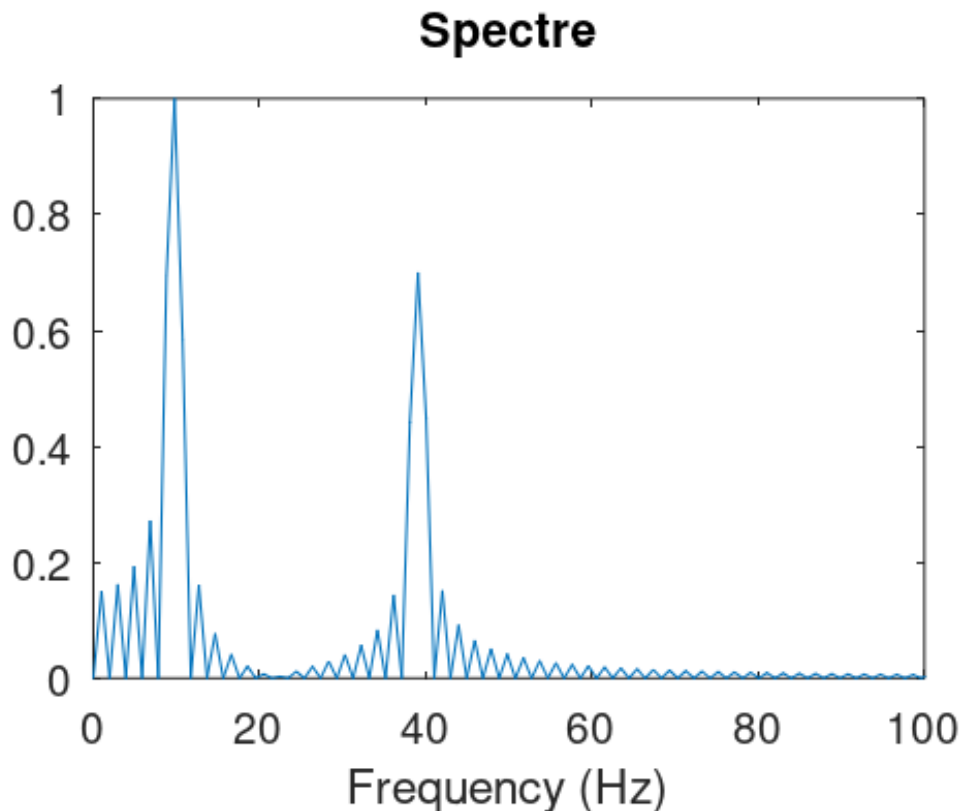


Рис. 2.21: Спектр суммарного сигнала

8. Выполнить задание с другой частотой дискретизации. Пояснить, что будет, если взять частоту дискретизации меньше 80 Гц?

Это приведет к нарушению теоремы Котельникова.

Наша максимальная частота – 40 Гц, это частота второго сигнала, - значит, частота дискретизации должна быть не менее 80 Гц (частота дискретизации должна быть минимум в два раза выше, чем максимальная частота в сигнале)

Если мы возьмем частоту дискретизации меньше 80 Гц, то произойдет наложение спектров. Более высокие частоты в сигнале будут “складываться” в более низкие частоты, искажая спектр и сам сигнал.

## **2.4 Амплитудная модуляция**

### **2.4.1 Постановка задачи**

Продемонстрировать принципы модуляции сигнала на примере аналоговой амплитудной модуляции

### **2.4.2 Порядок выполнения работы**

1. В вашем рабочем каталоге создайте каталог modulation и в нём новый сценарий с именем am.m.
2. Добавьте в файле am.m следующий код:(рис. 2.22).

```

1  % modulation/am.m
2  % Создание каталогов signal и spectre для размещения графиков:
3  mkdir 'signal';
4  mkdir 'spectre';
5  % Модуляция синусоид с частотами 50 и 5
6  % Длина сигнала (с)
7  tmax = 0.5;
8  % Частота дискретизации (Гц) (количество отсчетов)
9  fd = 512;
10 % Частота сигнала (Гц)
11 f1 = 5;
12 % Частота несущей (Гц)
13 f2 = 50;
14 % Спектр сигнала
15 fd2 = fd/2;
16 % Построение графиков двух сигналов (синусоиды)
17 % равной частоты
18 % Массив отсчетов времени:
19 t = 0:1./fd:tmax;
20 signal1 = sin(2*pi*t*f1);
21 signal2 = sin(2*pi*t*f2);
22 signal = signal1 .* signal2;
23 plot(signal, 'b');
24 hold on
25 % Построение отсбавшей:
26 plot(signal1, 'r');
27 plot(-signal1, 'r');
28 hold off
29 title('Signal');
30 print 'signal/am.png';
31 % Расчет спектра:
32 % Амплитуды преобразования Фурье-сигнала:
33 spectre = fft(signal,fd);
34 % Сетка частот:
35 f = 1000*(0:fd2)./(2*fd);
36 % Нормировка спектра по амплитуде:
37 spectre = 2*sqrt(spectre.*conj(spectre))./fd2;
38 % Построение спектра:
39 plot(f,spectre(1:fd2+1), 'b')
40 xlim([0 100]);
41
42 title('Spectre');
43 xlabel('Frequency (Hz)');

```

Рис. 2.22: Код am.m

В результате получаем, что спектр произведения представляет собой свёртку спектров(рис. 2.23),(рис. 2.24).

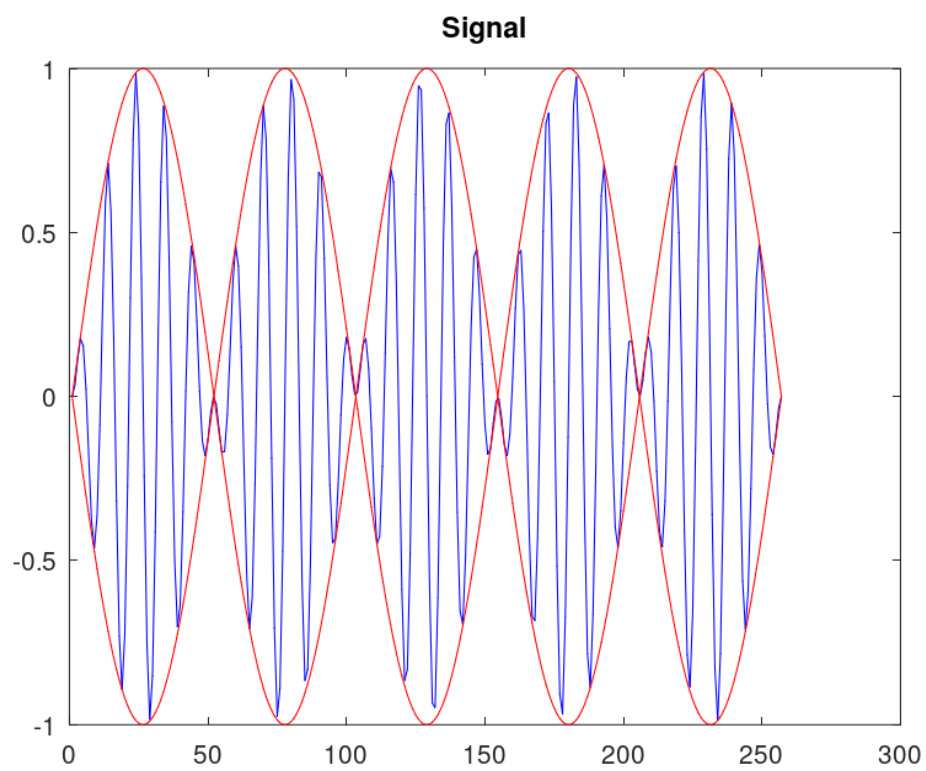


Рис. 2.23: Сигнал и огибающая при амплитудной модуляции

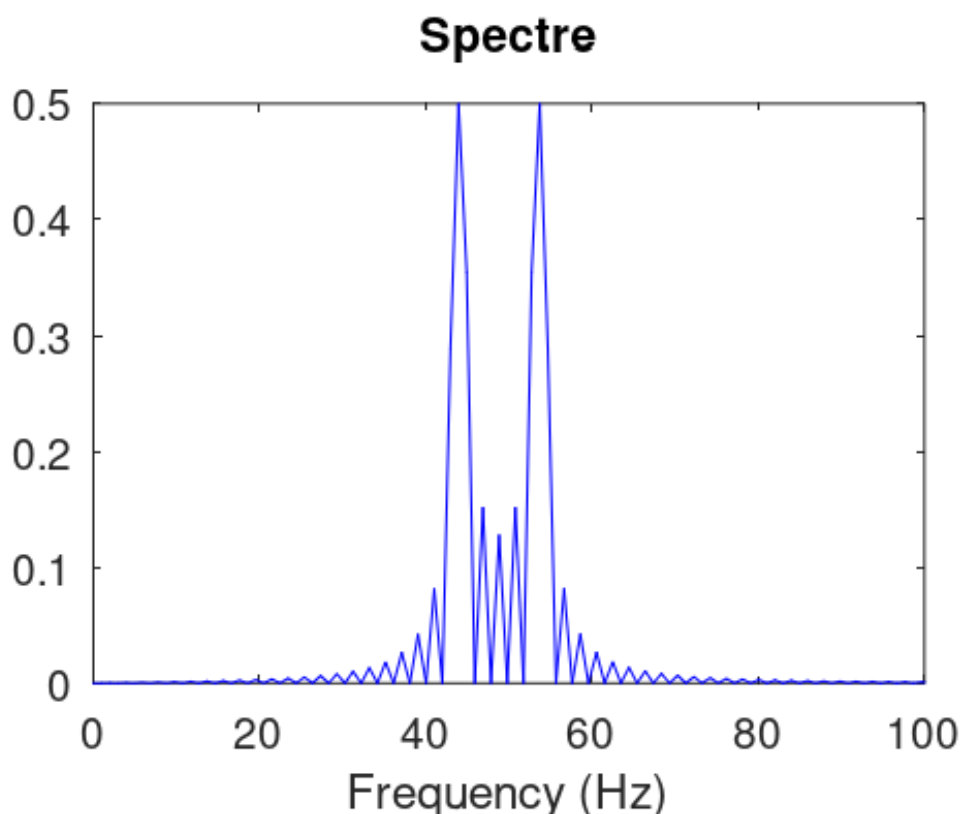


Рис. 2.24: Спектр сигнала при амплитудной модуляции

## 2.5 Кодирование сигнала. Исследование свойства самосинхронизации сигнала

### 2.5.1 Постановка задачи

По заданным битовым последовательностям требуется получить кодированные сигналы для нескольких кодов, проверить свойства самосинхронизуемости кодов, получить спектры.

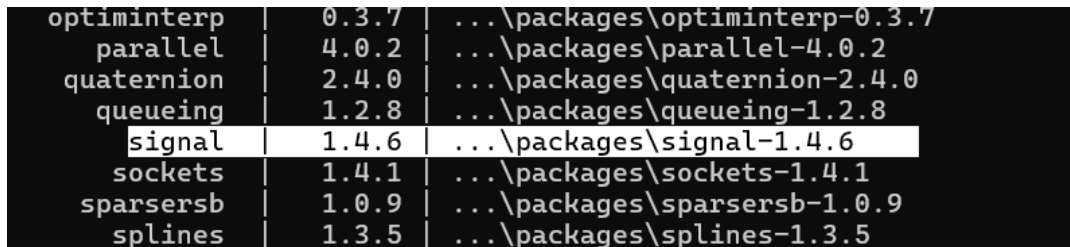
### 2.5.2 Порядок выполнения работы

1. В вашем рабочем каталоге создайте каталог coding и в нём файлы main.m, maptowave.m, unipolar.m, ami.m, bipolarnrz.m, bipolarrrz.m, manchester.m,



diffmanc.m, calcspectre.m.

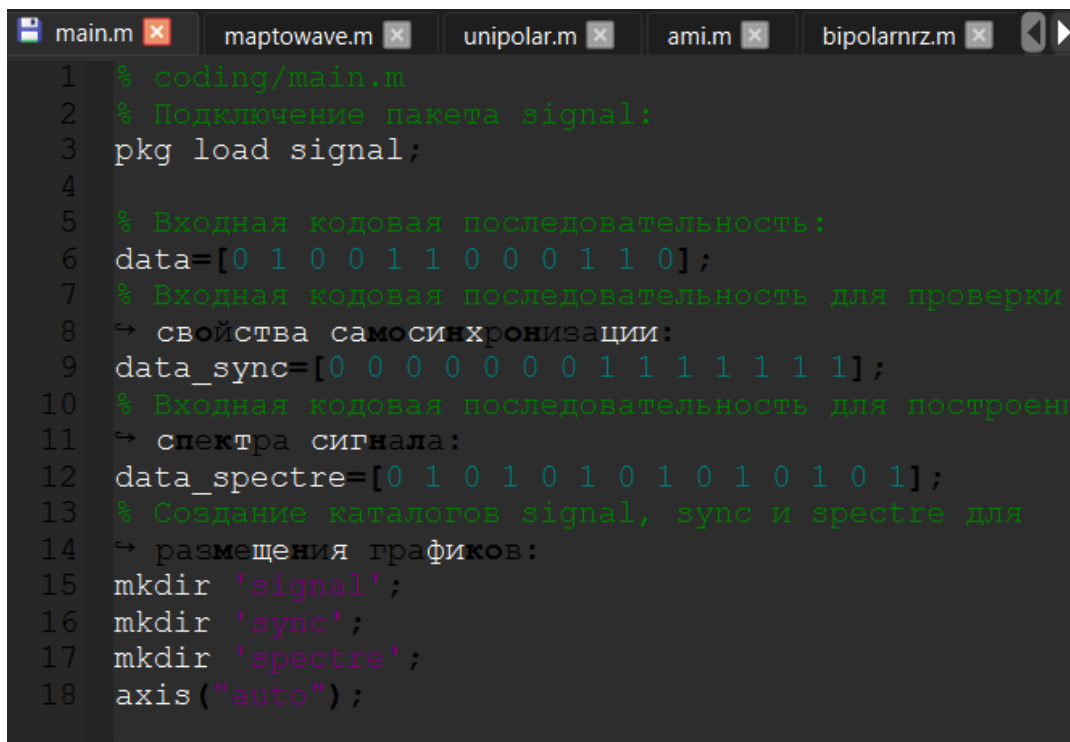
2. В окне интерпретатора команд проверьте, установлен ли у вас пакет расширений signal: (рис. 2.25).



optiminterp	0.3.7	...\packages\optiminterp-0.3.7
parallel	4.0.2	...\packages\parallel-4.0.2
quaternion	2.4.0	...\packages\quaternion-2.4.0
queueing	1.2.8	...\packages\queueing-1.2.8
signal	1.4.6	...\packages\signal-1.4.6
sockets	1.4.1	...\packages\sockets-1.4.1
sparsersb	1.0.9	...\packages\sparsersb-1.0.9
splines	1.3.5	...\packages\splines-1.3.5

Рис. 2.25: Окно терминала. Проверяем наличие signal

3. В файле main.m подключите пакет signal и задайте входные кодовые последовательности:(рис. 2.26).

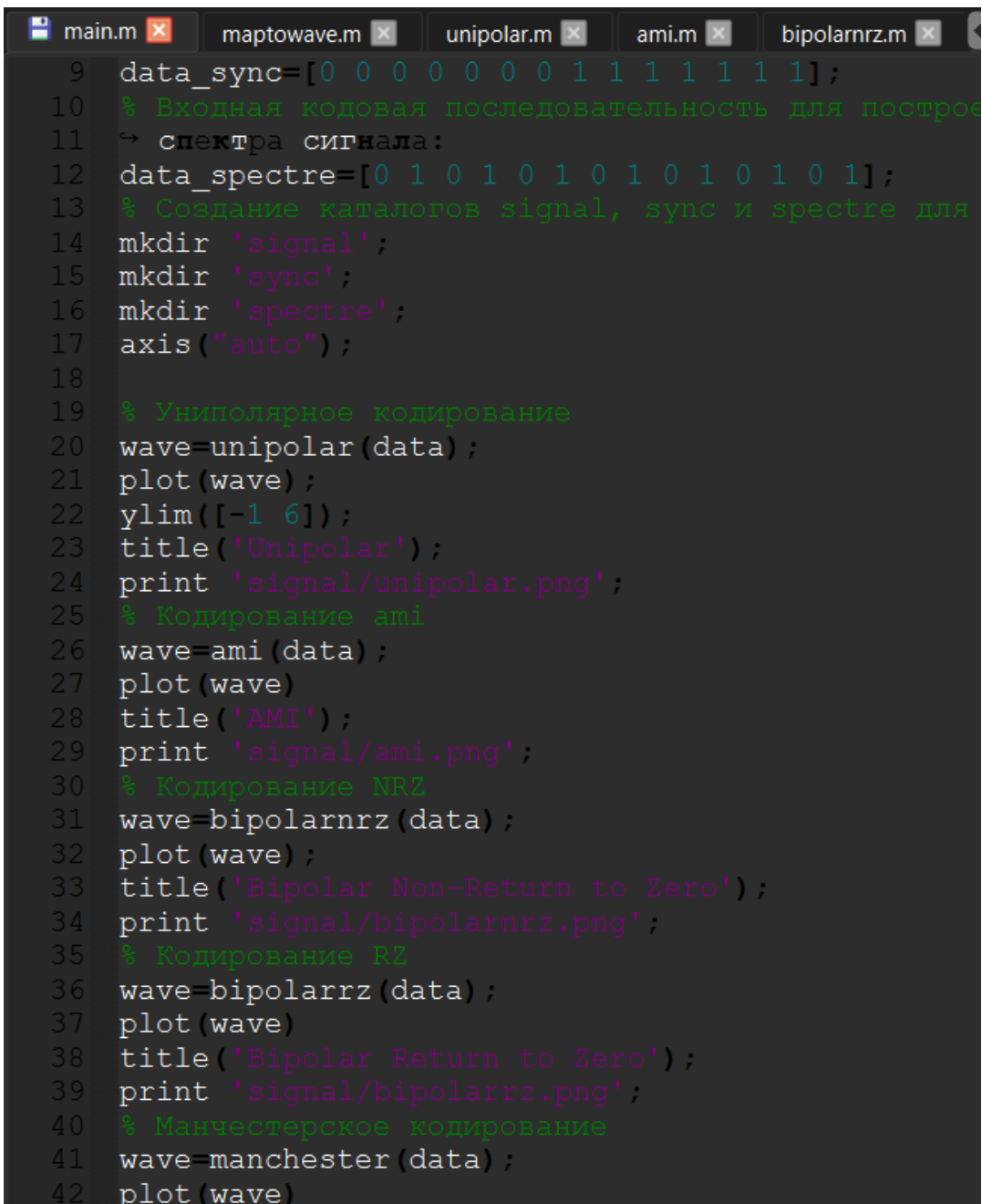


```
1 % coding/main.m
2 % Подключение пакета signal:
3 pkg load signal;
4
5 % Входная кодовая последовательность:
6 data=[0 1 0 0 1 1 0 0 0 1 1 0];
7 % Входная кодовая последовательность для проверки
8 % свойства самосинхронизации:
9 data_sync=[0 0 0 0 0 0 0 1 1 1 1 1];
10 % Входная кодовая последовательность для построения
11 % спектра сигнала:
12 data_spectre=[0 1 0 1 0 1 0 1 0 1 0 1];
13 % Создание каталогов signal, sync и spectre для
14 % размещения графиков:
15 mkdir 'signal';
16 mkdir 'sync';
17 mkdir 'spectre';
18 axis("auto");
```

Рис. 2.26: Код main.m

Затем в этом же файле пропишите вызовы функций для построения графиков

модуляций кодированных сигналов для кодовой последовательности data:(рис. 2.27).

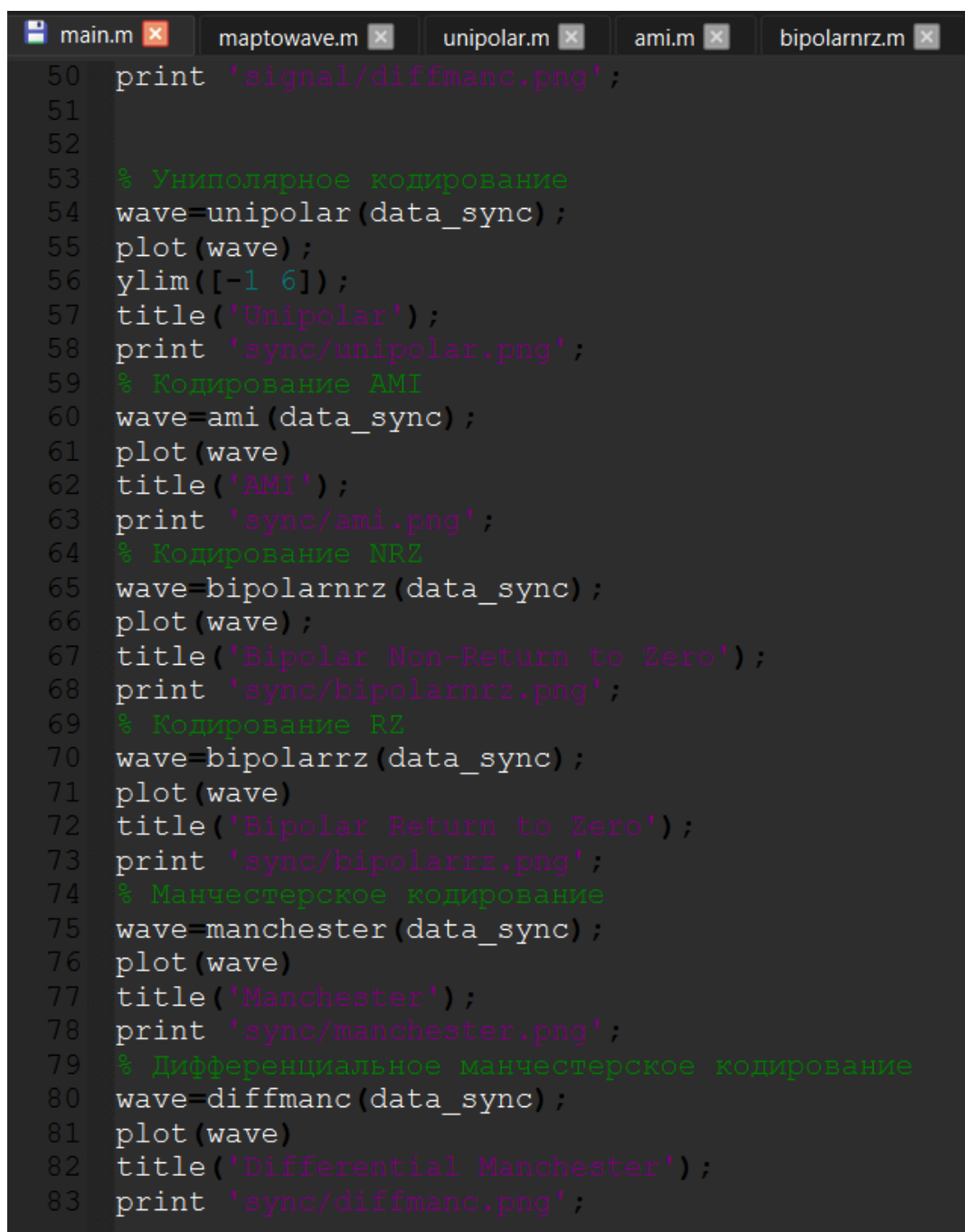


```
9 data_sync=[0 0 0 0 0 0 1 1 1 1 1 1];
10 % Входная кодовая последовательность для построения
11 % спектра сигнала:
12 data_spectre=[0 1 0 1 0 1 0 1 0 1 0 1];
13 % Создание каталогов signal, sync и spectre для
14 mkdir 'signal';
15 mkdir 'sync';
16 mkdir 'spectre';
17 axis("auto");
18
19 % Униполярное кодирование
20 wave=unipolar(data);
21 plot(wave);
22 ylim([-1 6]);
23 title('Unipolar');
24 print 'signal/unipolar.png';
25 % Кодирование аml
26 wave=ami(data);
27 plot(wave);
28 title('AMI');
29 print 'signal/ami.png';
30 % Кодирование NRZ
31 wave=bipolarnrz(data);
32 plot(wave);
33 title('Bipolar Non-Return to Zero');
34 print 'signal/bipolarnrz.png';
35 % Кодирование RZ
36 wave=bipolarrz(data);
37 plot(wave);
38 title('Bipolar Return to Zero');
39 print 'signal/bipolarrz.png';
40 % Манчестерское кодирование
41 wave=manchester(data);
42 plot(wave);
```

Рис. 2.27: Код main.m

Затем в этом же файле пропишите вызовы функций для построения графиков модуляций кодированных сигналов для кодовой последовательности

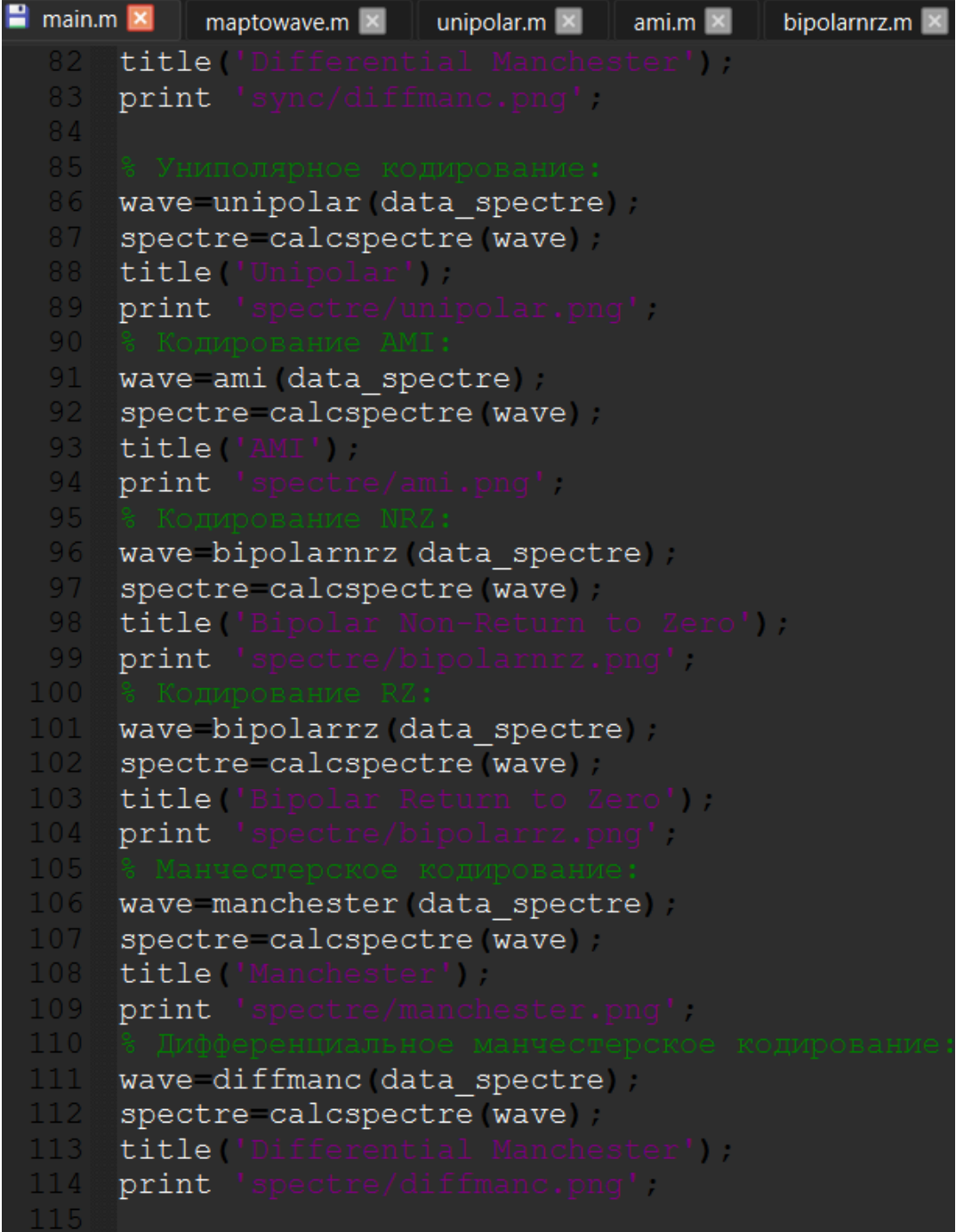
data\_sync:(рис. 2.28).



```
50 print 'signal/diffmanc.png';
51
52
53 % Униполярное кодирование
54 wave=unipolar(data_sync);
55 plot(wave);
56 ylim([-1 6]);
57 title('Unipolar');
58 print 'sync/unipolar.png';
59 % Кодирование AMI
60 wave=ami(data_sync);
61 plot(wave);
62 title('AMI');
63 print 'sync/ami.png';
64 % Кодирование NRZ
65 wave=bipolarnrz(data_sync);
66 plot(wave);
67 title('Bipolar Non-Return to Zero');
68 print 'sync/bipolarnrz.png';
69 % Кодирование RZ
70 wave=bipolarrrz(data_sync);
71 plot(wave);
72 title('Bipolar Return to Zero');
73 print 'sync/bipolarrrz.png';
74 % Манчестерское кодирование
75 wave=manchester(data_sync);
76 plot(wave);
77 title('Manchester');
78 print 'sync/manchester.png';
79 % Дифференциальное манчестерское кодирование
80 wave=diffmanc(data_sync);
81 plot(wave);
82 title('Differential Manchester');
83 print 'sync/diffmanc.png';
```

Рис. 2.28: Код main.m

Далее в этом же файле пропишите вызовы функций для построения графиков спектров:(рис. 2.29).



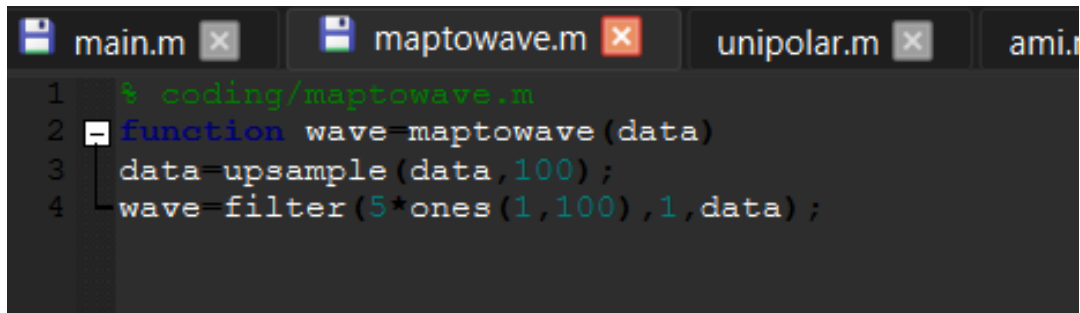
```

82 title('Differential Manchester');
83 print 'spectre/diffmanc.png';
84
85 % Униполярное кодирование:
86 wave=unipolar(data_spectre);
87 spectre=calcspectre(wave);
88 title('Unipolar');
89 print 'spectre/unipolar.png';
90 % Кодирование AMI:
91 wave=ami(data_spectre);
92 spectre=calcspectre(wave);
93 title('AMI');
94 print 'spectre/ami.png';
95 % Кодирование NRZ:
96 wave=bipolarnrz(data_spectre);
97 spectre=calcspectre(wave);
98 title('Bipolar Non-Return to Zero');
99 print 'spectre/bipolarnrz.png';
100 % Кодирование RZ:
101 wave=bipolarrz(data_spectre);
102 spectre=calcspectre(wave);
103 title('Bipolar Return to Zero');
104 print 'spectre/bipolarrz.png';
105 % Манчестерское кодирование:
106 wave=manchester(data_spectre);
107 spectre=calcspectre(wave);
108 title('Manchester');
109 print 'spectre/manchester.png';
110 % Дифференциальное манчестерское кодирование:
111 wave=diffmanc(data_spectre);
112 spectre=calcspectre(wave);
113 title('Differential Manchester');
114 print 'spectre/diffmanc.png';
115

```

Рис. 2.29: Код main.m

4. В файле maptowave.m пропишите функцию, которая по входному битовому потоку строит график сигнала:(рис. 2.30).

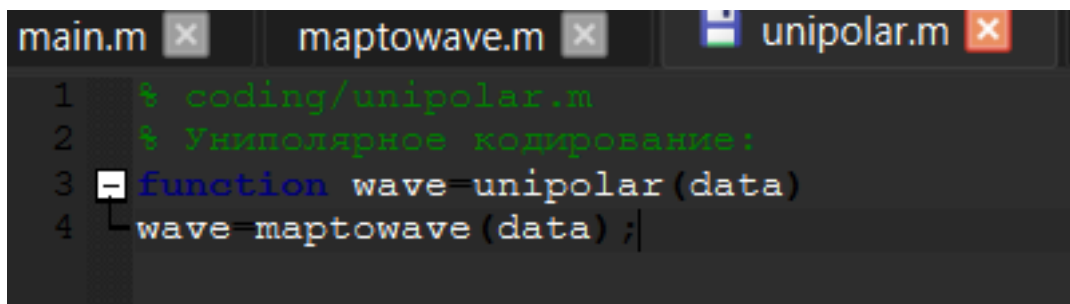
A screenshot of a MATLAB code editor window. The title bar shows four tabs: 'main.m', 'maptowave.m' (active), 'unipolar.m', and 'ami.m'. The code in the active tab is as follows:

```
1 % coding/maptowave.m
2 function wave=maptowave(data)
3 data=upsample(data,100);
4 wave=filter(5*ones(1,100),1,data);
```

Рис. 2.30: Код maptowave.m

5. В файлах unipolar.m, ami.m, bipolarnrz.m, bipolarrrz.m, manchester.m, diffmanc.m пропишите соответствующие функции преобразования кодовой последовательности data с вызовом функции maptowave для построения соответствующего графика.

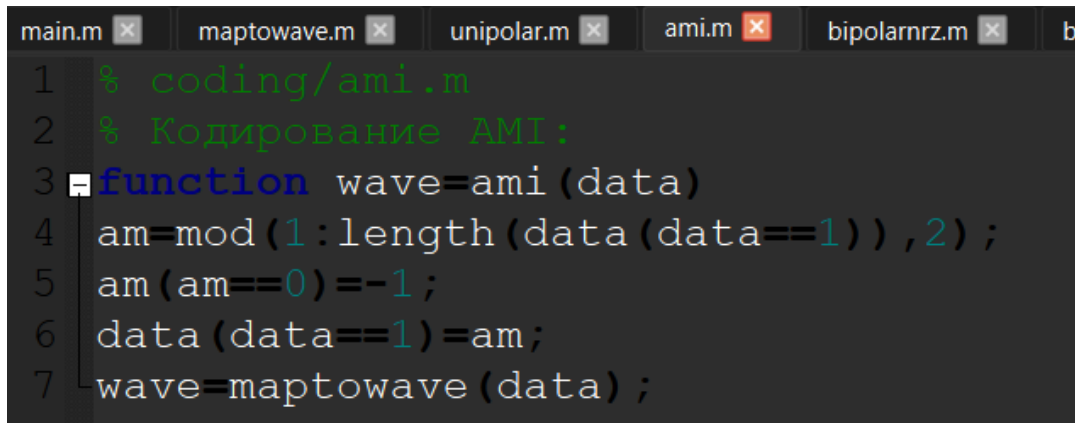
Униполярное кодирование:(рис. 2.31).

A screenshot of a MATLAB code editor window. The title bar shows three tabs: 'main.m', 'maptowave.m', and 'unipolar.m' (active). The code in the active tab is as follows:

```
1 % coding/unipolar.m
2 % Униполярное кодирование:
3 function wave=unipolar(data)
4 wave=maptowave(data);
```

Рис. 2.31: Код unipolar.m

Кодирование AMI:(рис. 2.32).



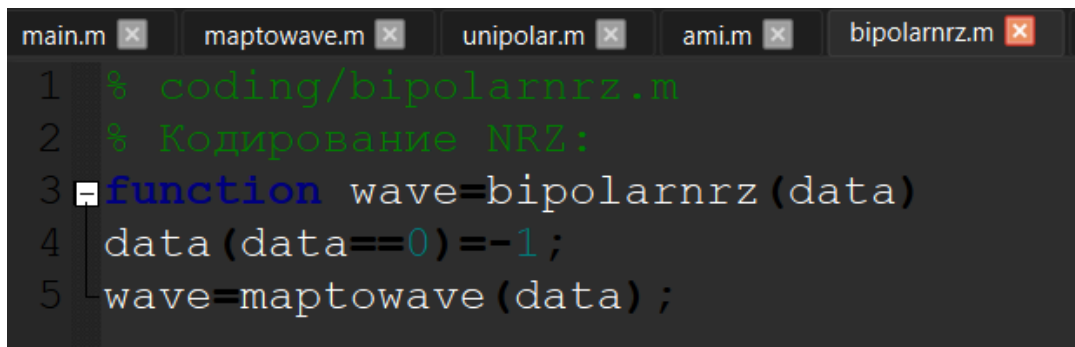
```

main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m x b
1 % coding/ami.m
2 % Кодирование AMI:
3 function wave=ami(data)
4 am=mod(1:length(data(data==1)),2);
5 am(am==0)=-1;
6 data(data==1)=am;
7 wave=maptowave(data);

```

Рис. 2.32: Код ami.m

Кодирование NRZ:(рис. 2.33).



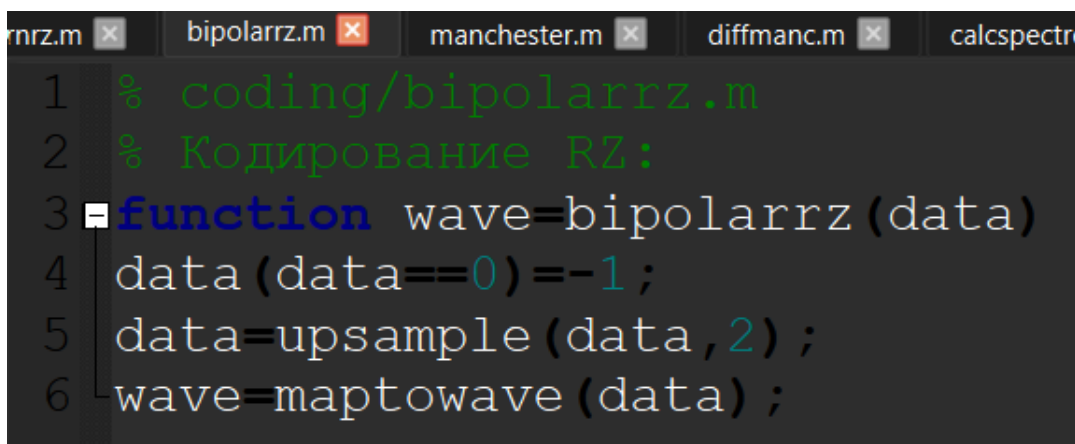
```

main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m x
1 % coding/bipolarnrz.m
2 % Кодирование NRZ:
3 function wave=bipolarnrz(data)
4 data(data==0)=-1;
5 wave=maptowave(data);

```

Рис. 2.33: Код bipolarnrz.m

Кодирование RZ:(рис. 2.34).



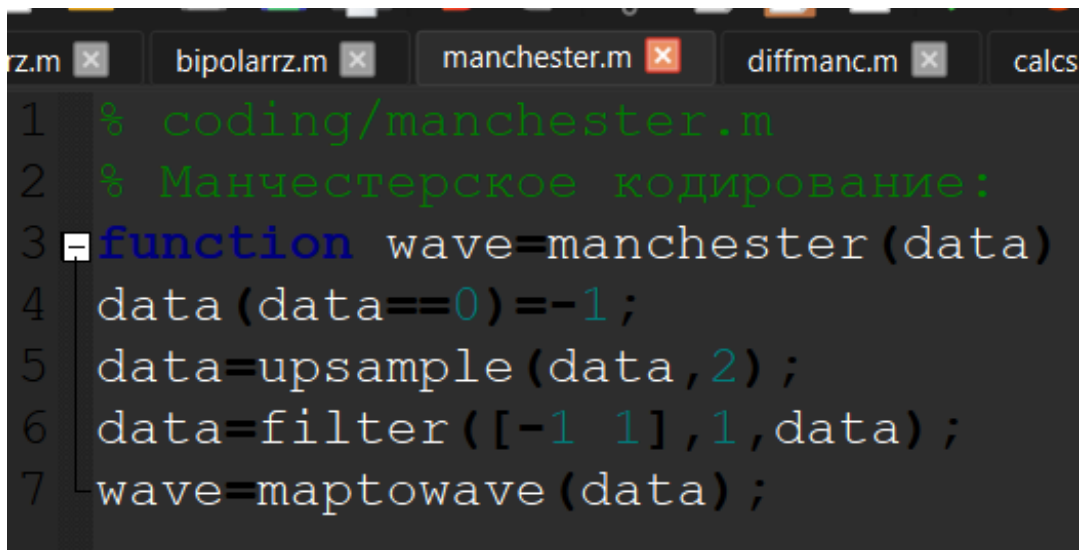
```

rnrz.m x bipolarrrz.m x manchester.m x diffmanc.m x calcspectre
1 % coding/bipolarrrz.m
2 % Кодирование RZ:
3 function wave=bipolarrrz(data)
4 data(data==0)=-1;
5 data=upsample(data,2);
6 wave=maptowave(data);

```

Рис. 2.34: Код bipolarrrz.m

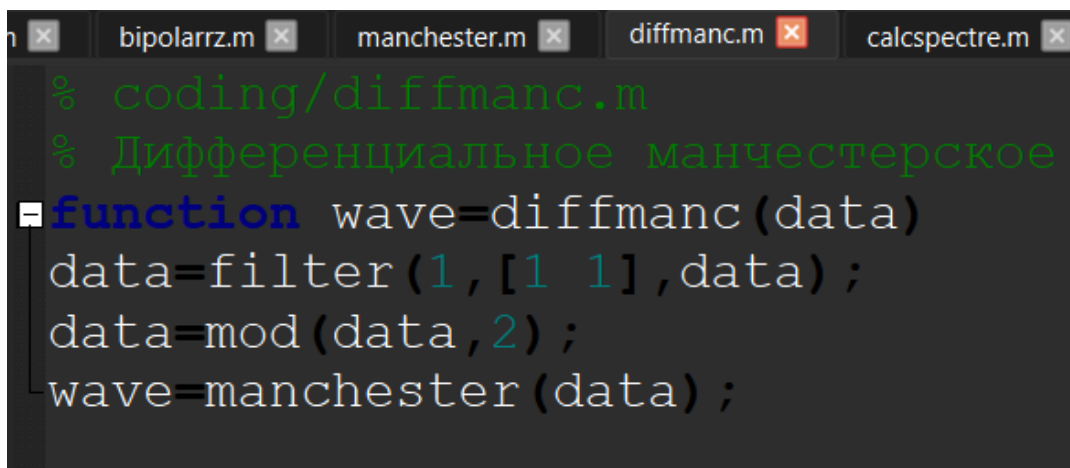
Манчестерское кодирование:(рис. 2.35).

A screenshot of a MATLAB script editor showing the code for the 'manchester.m' file. The editor has several tabs at the top: 'z.m', 'bipolarrz.m', 'manchester.m' (which is the active tab and has a red 'x' icon), 'diffmanc.m', and 'calcs'. The code is as follows:

```
1 % coding/manchester.m
2 % Манчестерское кодирование:
3 function wave=manchester(data)
4 data(data==0)=-1;
5 data=upsample(data,2);
6 data=filter([-1 1],1,data);
7 wave=maptowave(data);
```

Рис. 2.35: Код manchester.m

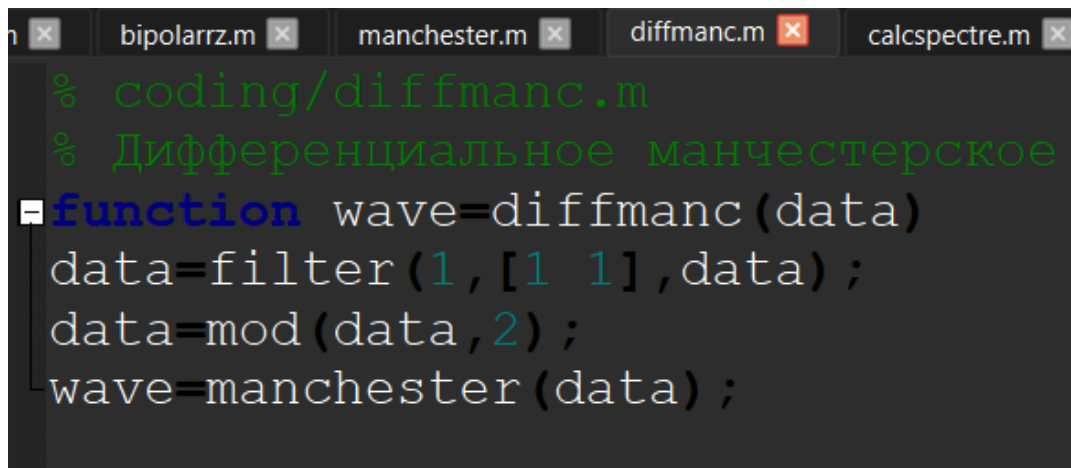
Дифференциальное манчестерское кодирование:(рис. 2.36).

A screenshot of a MATLAB script editor showing the code for the 'diffmanc.m' file. The editor has several tabs at the top: 'h', 'bipolarrz.m', 'manchester.m', 'diffmanc.m' (which is the active tab and has a red 'x' icon), and 'calcspectre.m'. The code is as follows:

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

Рис. 2.36: Код diffmanc.m

6. В файле calcspectre.m пропишите функцию построения спектра сигнала:(рис. 2.37).

The image shows a MATLAB code editor window with several tabs at the top: 'bipolarrz.m', 'manchester.m', 'diffmanc.m' (which is the active tab), and 'calcspectre.m'. The code in the active tab is as follows:

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

Рис. 2.37: Код calcspectre.m

7. Запустите главный скрипт main.m. В каталоге signal должны быть получены файлы с графиками кодированного сигнала, в каталоге sync — файлы с графиками, иллюстрирующими свойства самосинхронизации, в каталоге spectre — файлы с графиками спектров сигналов. (рис. 2.38), (рис. 2.39), (рис. 2.40), (рис. 2.41), (рис. 2.42), (рис. 2.43), (рис. 2.44), (рис. 2.45), (рис. 2.46), (рис. 2.47), (рис. 2.48), (рис. 2.49), (рис. 2.50), (рис. 2.51), (рис. 2.52), (рис. 2.53), (рис. 2.54), (рис. 2.55).



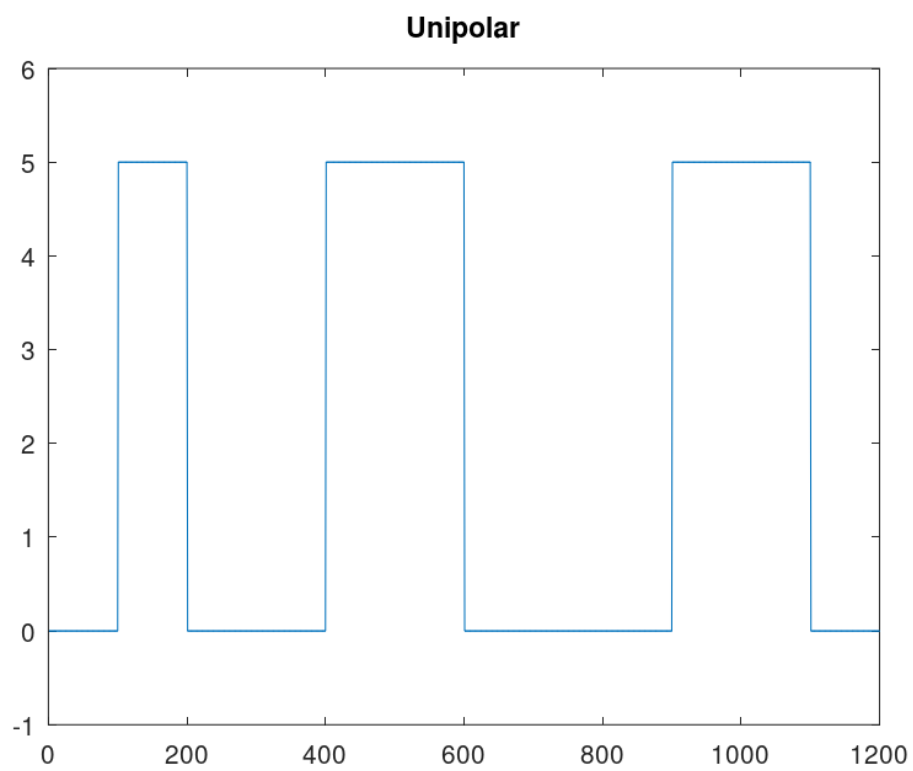


Рис. 2.38: Униполярное кодирование

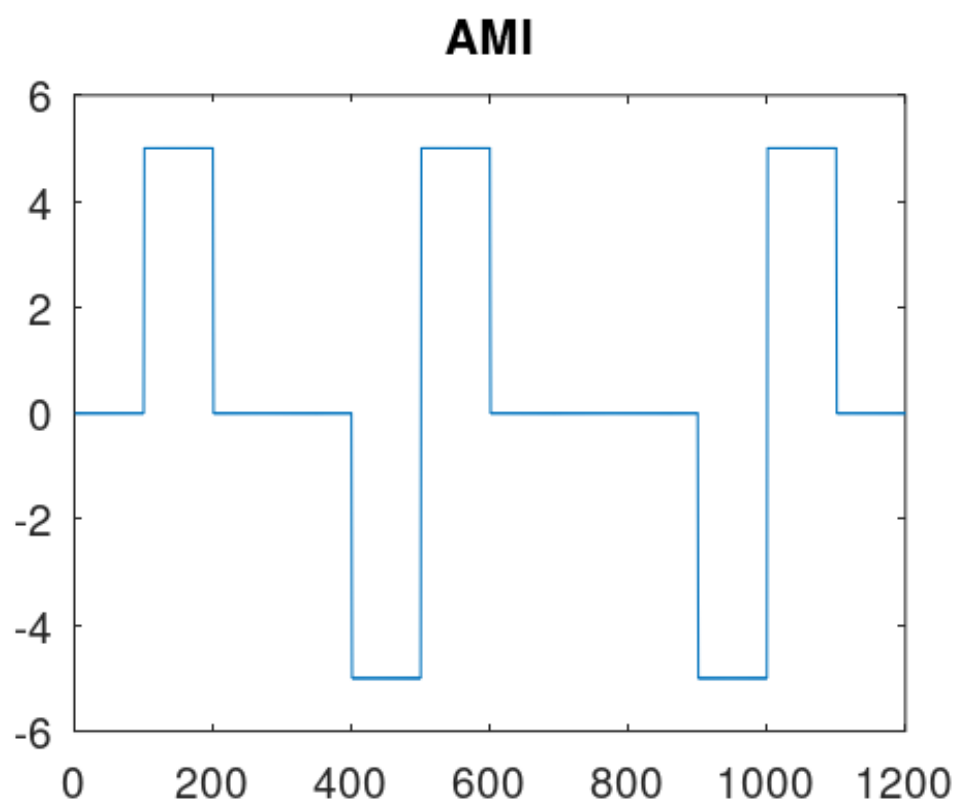


Рис. 2.39: Кодирование AMI

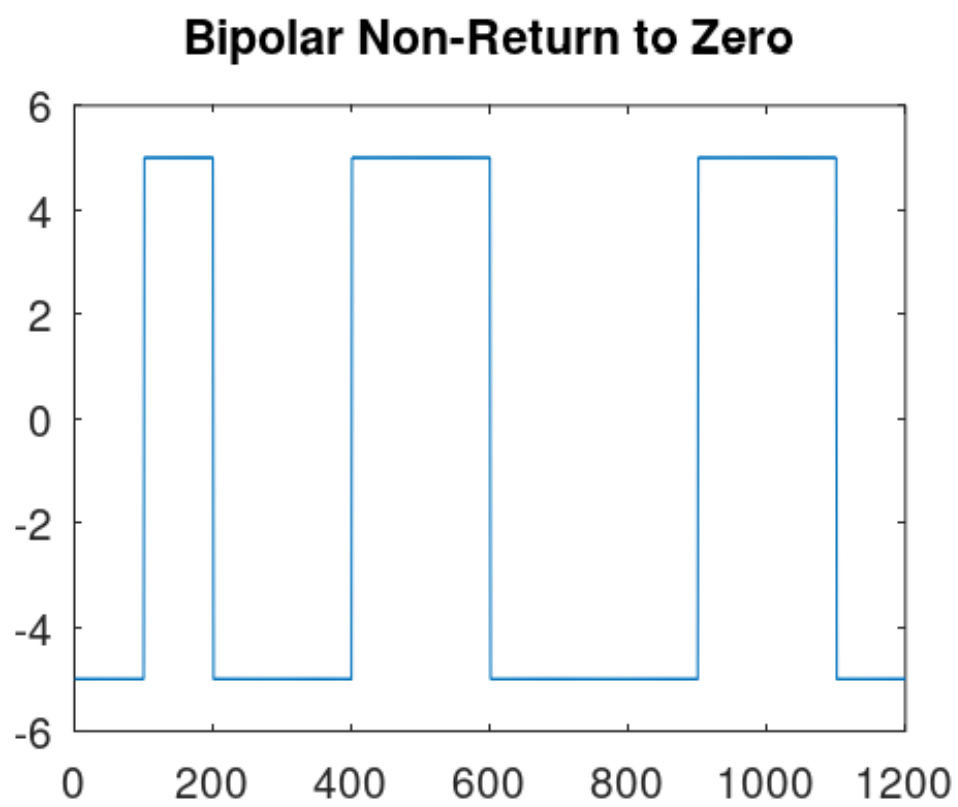
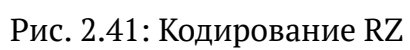


Рис. 2.40: Кодирование NRZ



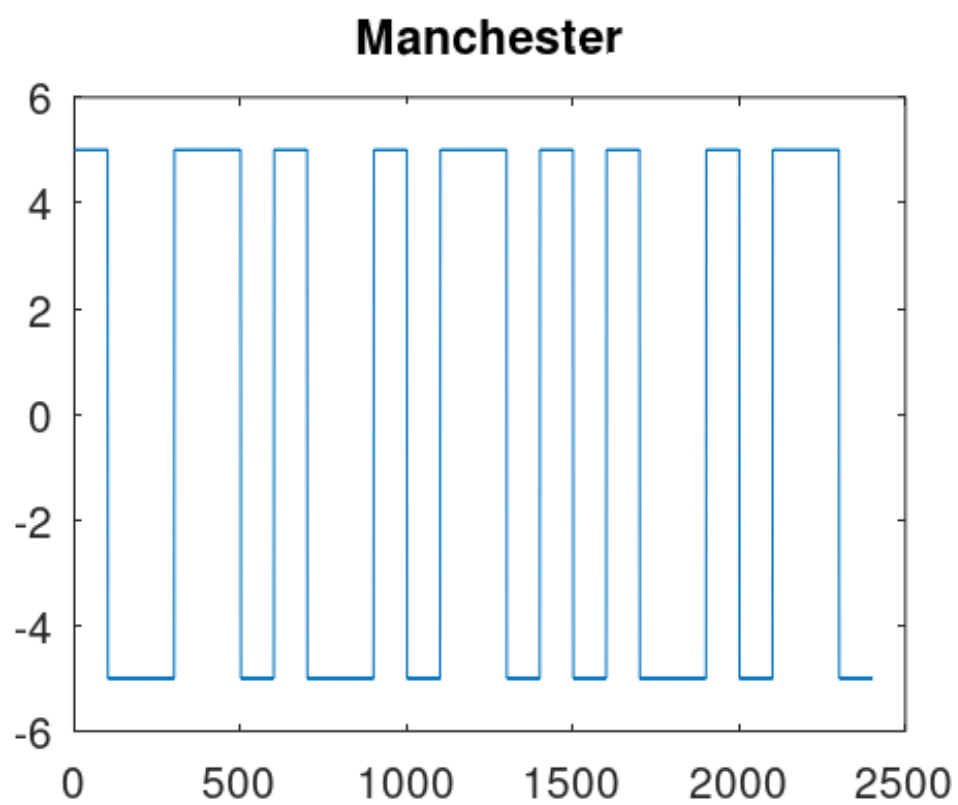


Рис. 2.42: Манчестерское кодирование

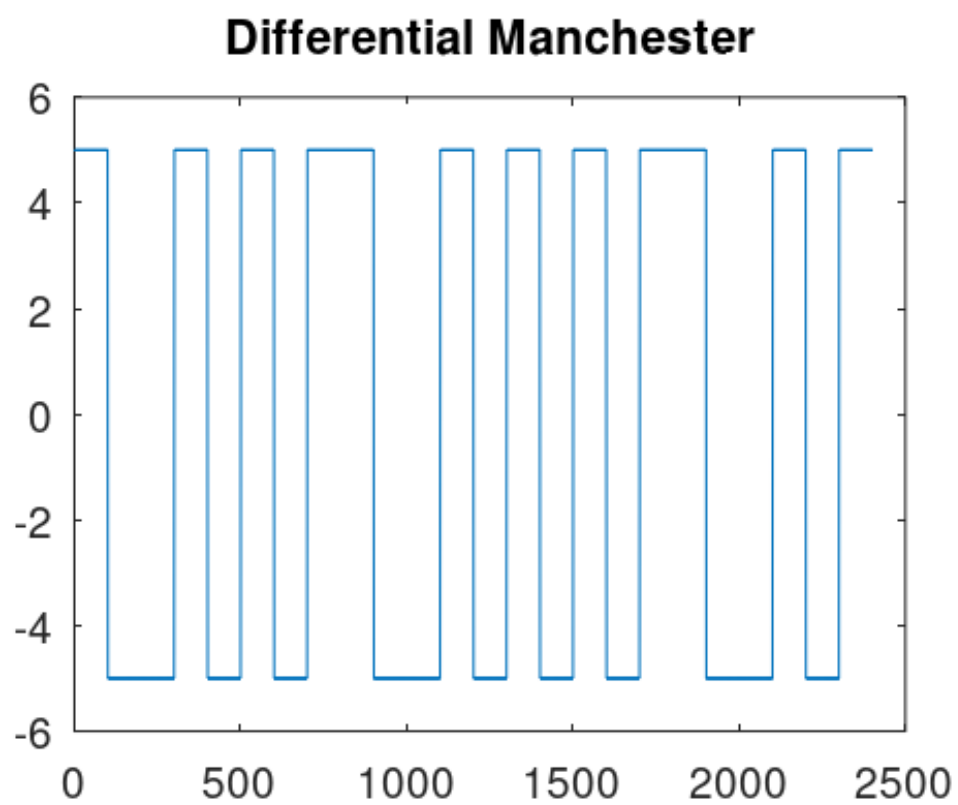


Рис. 2.43: Дифференциальное манчестерское кодирование

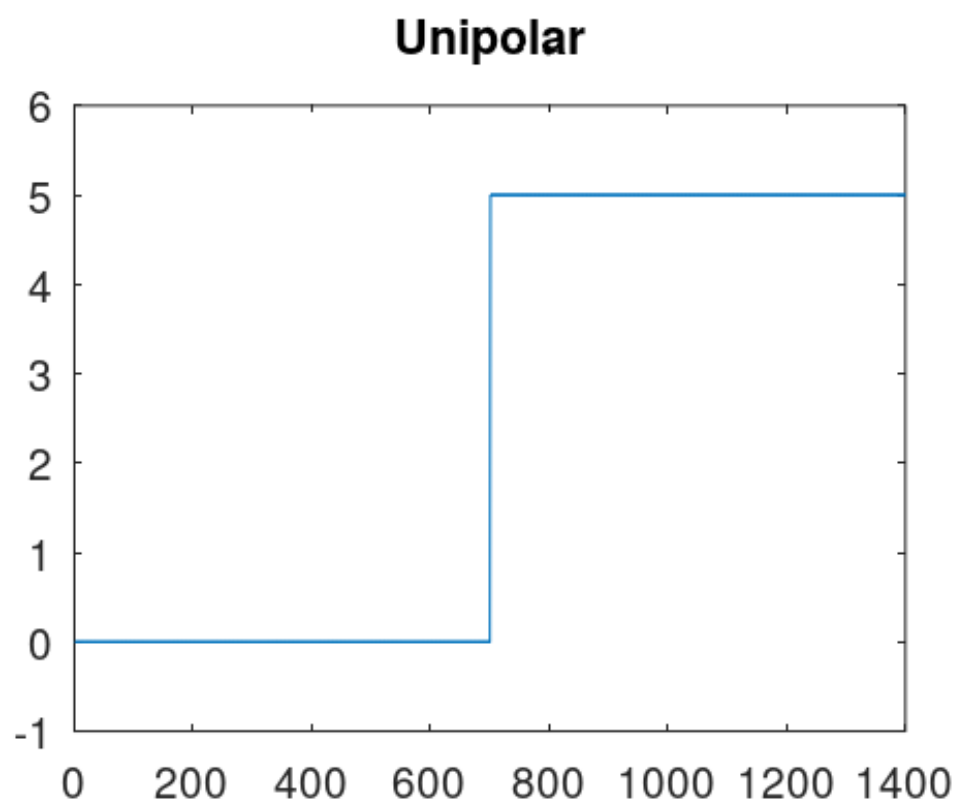


Рис. 2.44: Униполярное кодирование: нет самосинхронизации

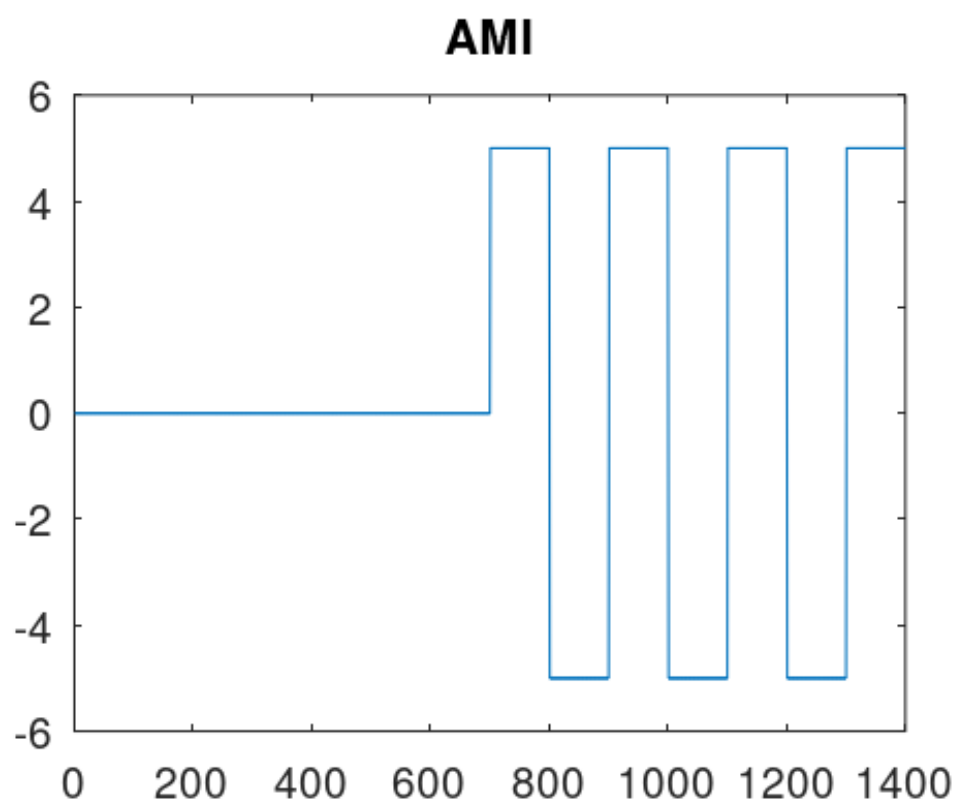


Рис. 2.45: Кодирование AMI: самосинхронизация при наличии сигнала



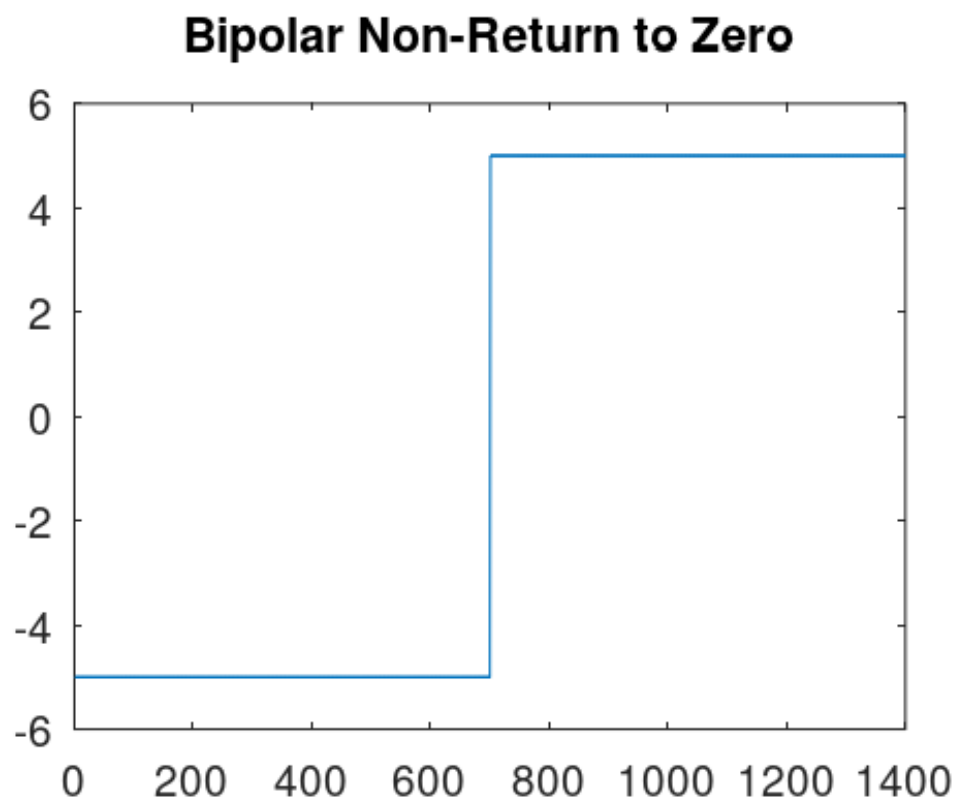


Рис. 2.46: Кодирование NRZ: нет самосинхронизации

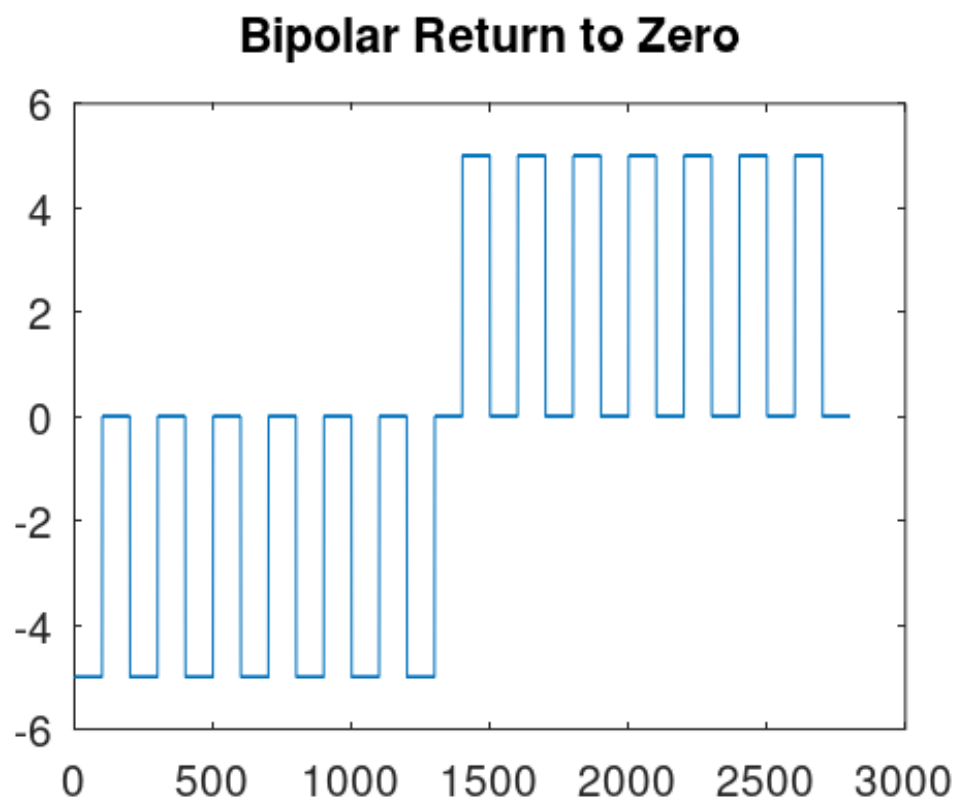


Рис. 2.47: Кодирование RZ: есть самосинхронизация

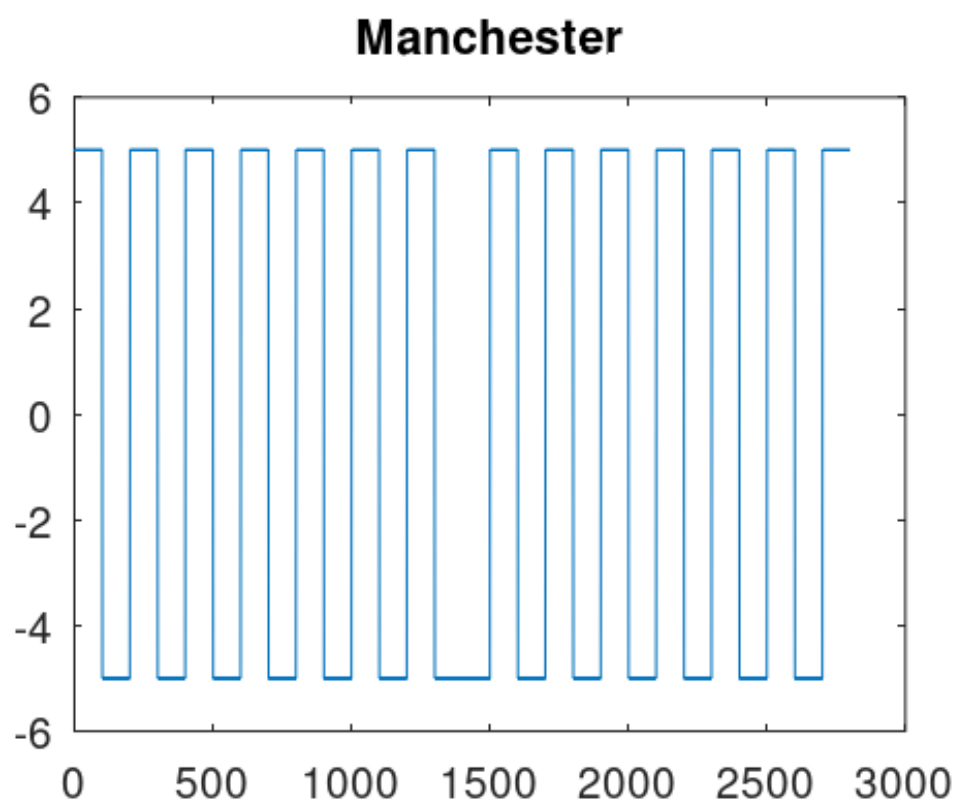


Рис. 2.48: Манчестерское кодирование: есть самосинхронизация

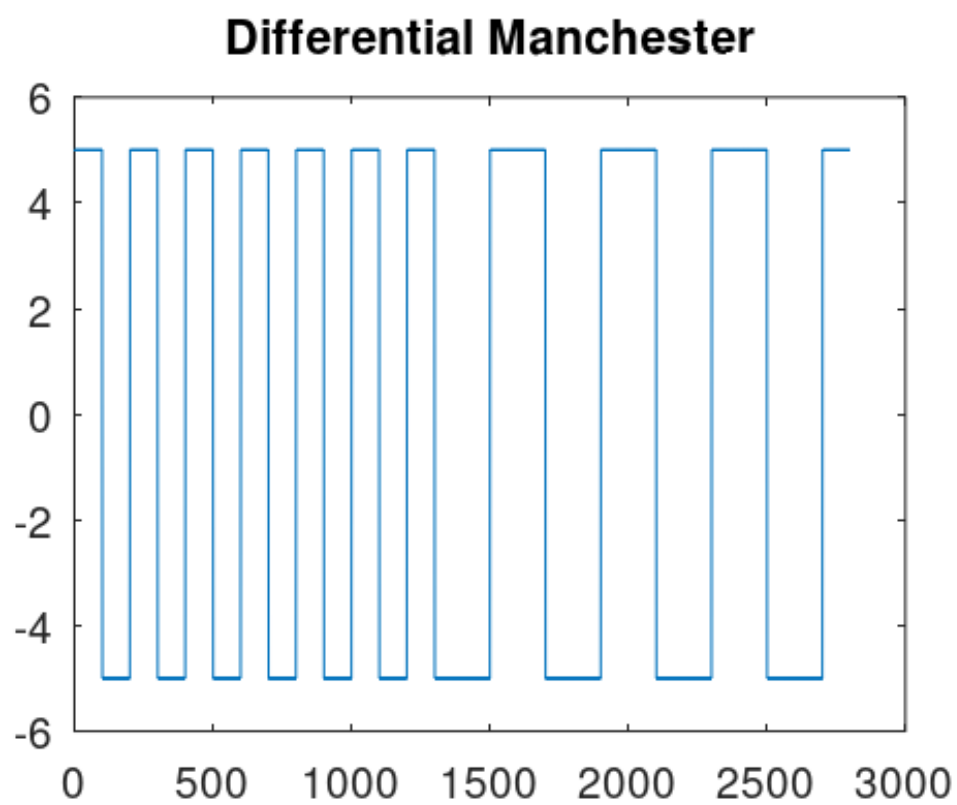


Рис. 2.49: Дифференциальное манчестерское кодирование: есть самосинхронизация

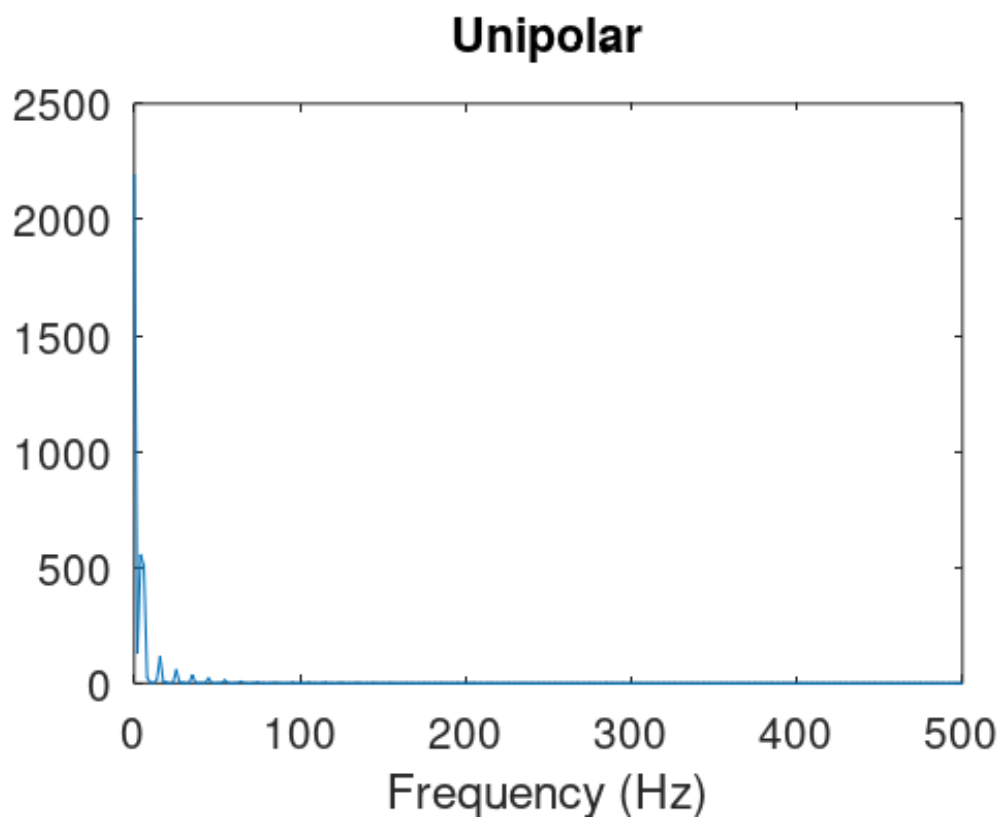


Рис. 2.50: Униполярное кодирование: спектр сигнала

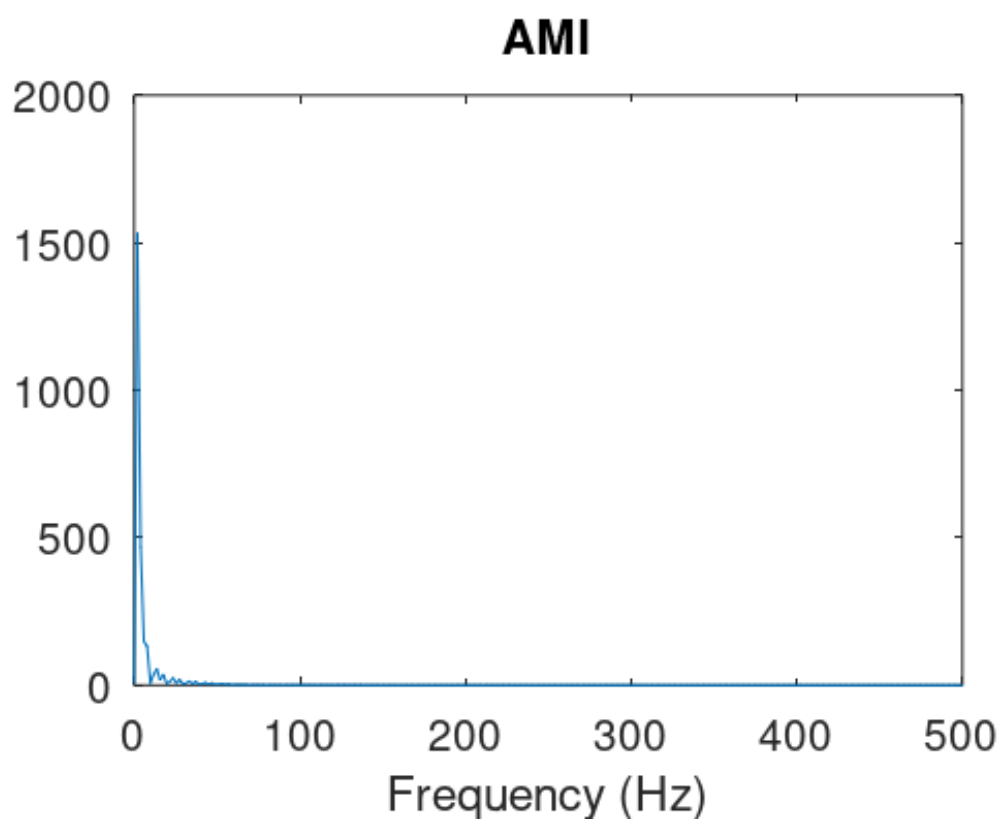


Рис. 2.51: Кодирование AMI: спектр сигнала

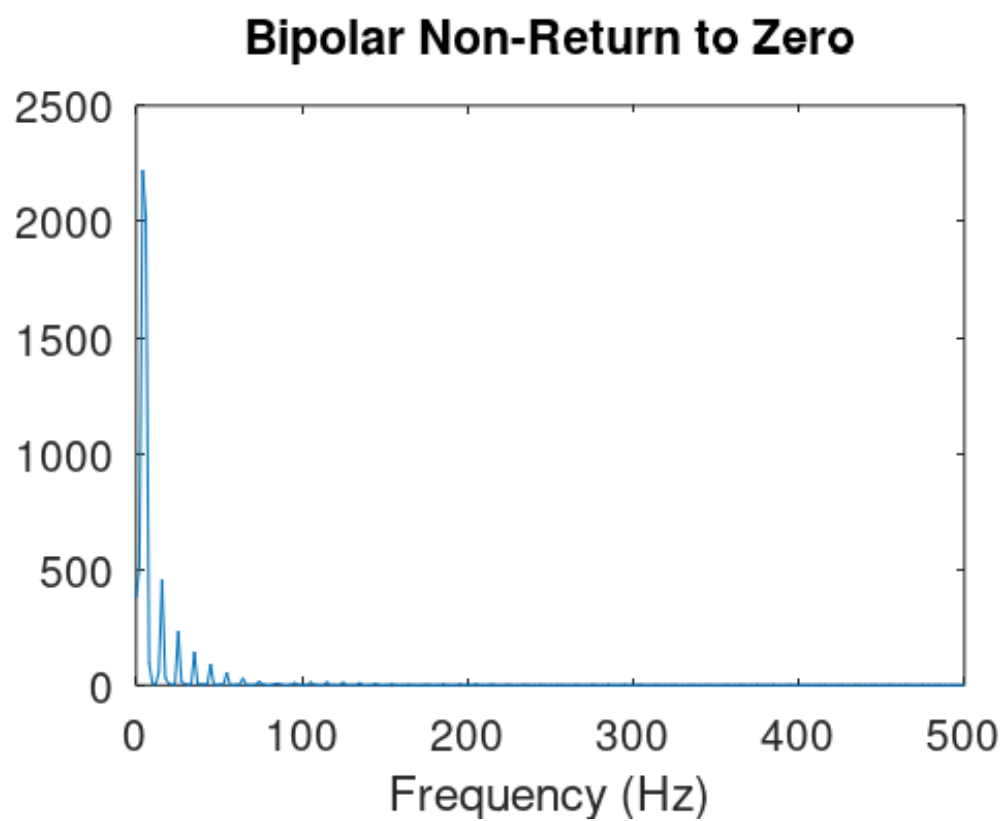


Рис. 2.52: Кодирование NRZ: спектр сигнала

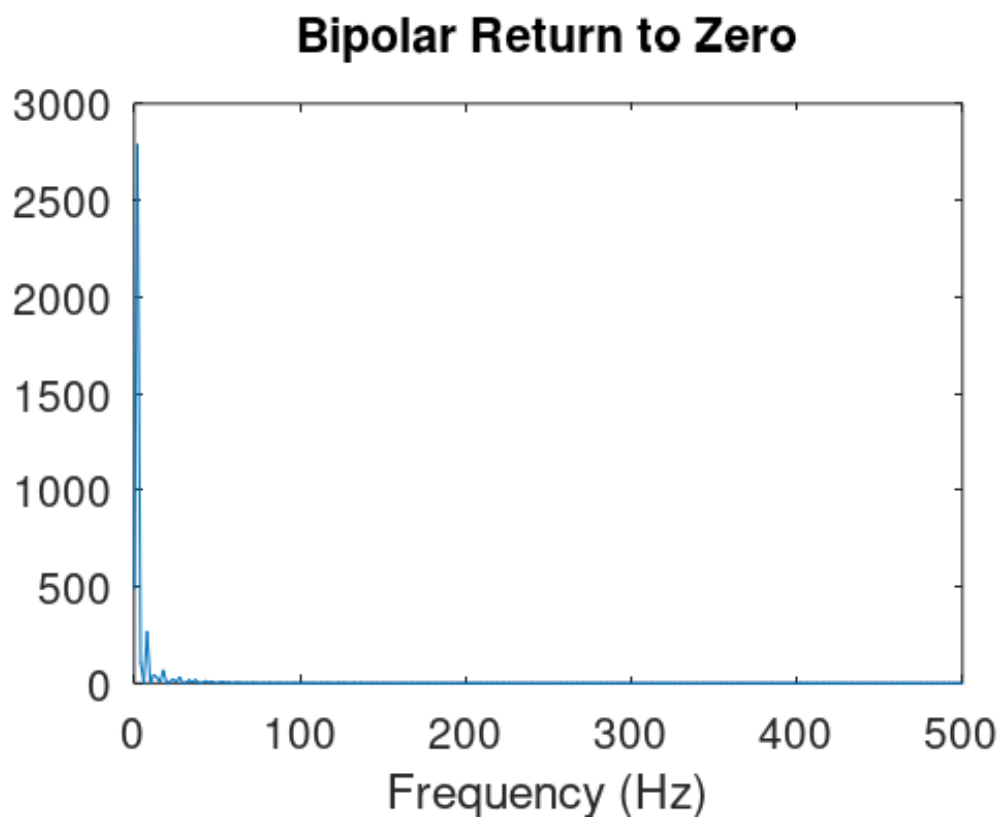


Рис. 2.53: Кодирование RZ: спектр сигнала



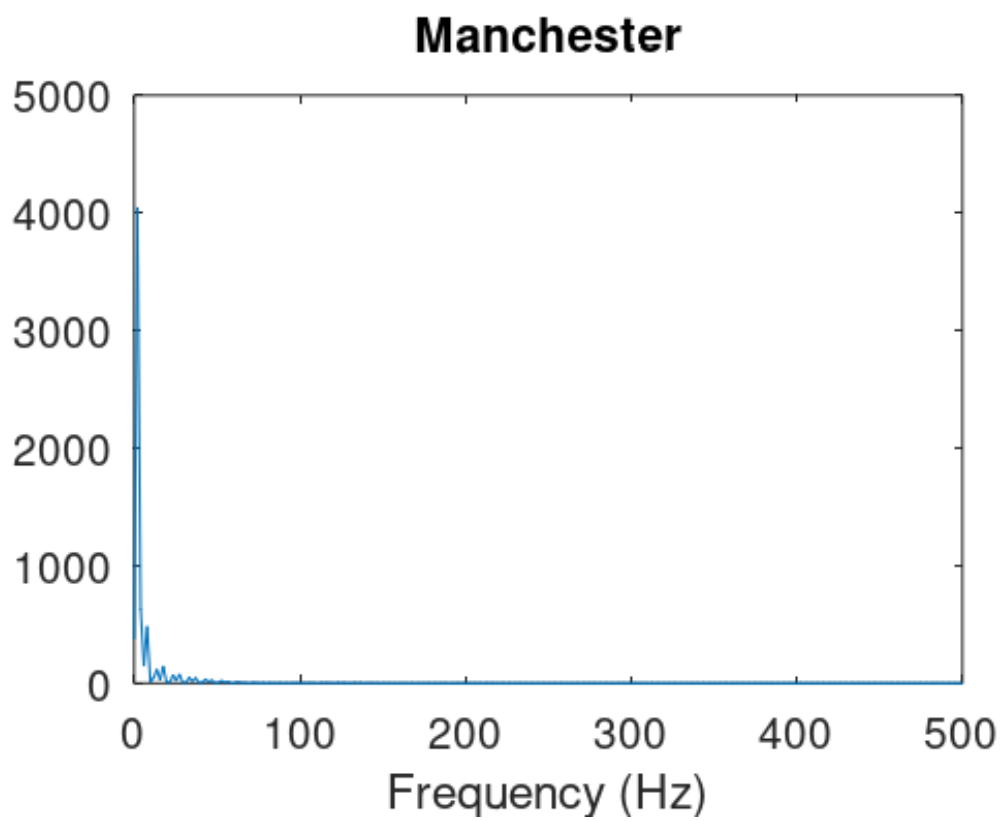


Рис. 2.54: Манчестерское кодирование: спектр сигнала

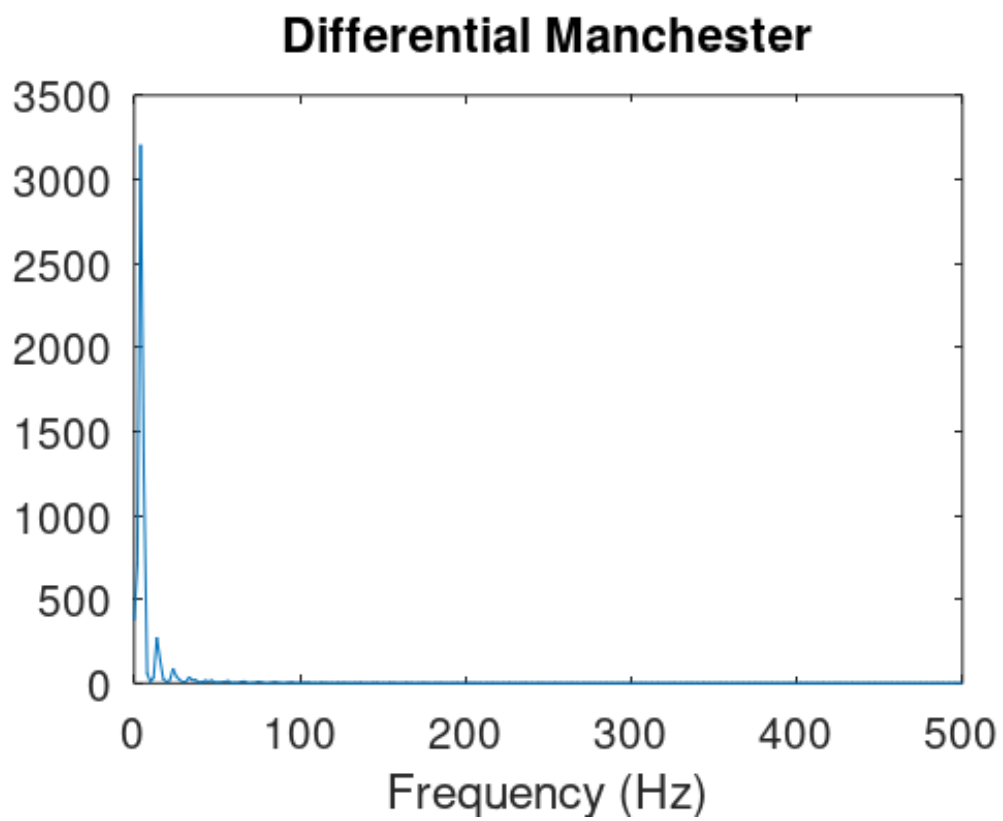


Рис. 2.55: Дифференциальное манчестерское кодирование: спектр сигнала

## 3 Выводы

Изучили методы кодирования и модуляции сигналов с помощью высокоуровневого языка программирования Octave. Поняли определения спектра и параметров сигнала. Продемонстрировали понимание принципов модуляции сигнала на примере аналоговой амплитудной модуляции. Исследовали свойства самосинхронизации сигнала.

## **Список литературы**