

Отчёт по лабораторной работе №1

Дисциплина: Сетевые технологии

Мишина Анастасия Алексеевна

Содержание

1	Цель работы	6
2	Выполнение лабораторной работы	7
2.1	Построение графиков в Octave	7
2.2	Разложение импульсного сигнала в частичный ряд Фурье .	10
2.3	Определение спектра и параметров сигнала	14
2.4	Амплитудная модуляция	21
2.5	Кодирование сигнала. Исследование свойства самосинхронизации сигнала	24
3	Выводы	40

Список иллюстраций

2.1	Листинг файла plot_sin.m	7
2.2	График функций y1 на интервале $-10; 10$	8
2.3	Файлы .eps, .png	8
2.4	Листинг файла plot_sin_cos.m	9
2.5	График функций y1 и y2 на интервале $-10; 10$	9
2.6	Листинг файла meandr.m	11
2.7	Меандр через косинусы	12
2.8	Листинг файла meandr.m	13
2.9	Меандр через синусы	14
2.10	Листинг файла spectre.m	15
2.11	Графики сигналов разной частоты	16
2.12	Листинг файла spectre.m	17
2.13	График спектра синусоидальных сигналов	18
2.14	Исправленный график спектров синусоидальных сигналов	18
2.15	Листинг файла spectre_sum.m	19
2.16	Суммарный сигнал	20
2.17	Спектр суммарного сигнала	21
2.18	Листинг файла am.m	22
2.19	Сигнал и огибающая при амплитудной модуляции	23
2.20	Спектр сигнала при амплитудной модуляции	23
2.21	Проверка правильности установки пакета signal	24
2.22	Задаем входные кодовые последовательности	24
2.23	Вызовы функций для посторения модуляций кодированных сигналов кодовой последовательности data	25
2.24	Вызовы функций для посторения модуляций кодированных сигналов кодовой последовательности data_sync	26
2.25	Вызовы функций для посторения графиков спектров	27
2.26	Листинг файла maptowave.m	28
2.27	Листинг файла unipolar.m	28
2.28	Листинг файла ami.m	28
2.29	Листинг файла bipolarnrz.m	29
2.30	Листинг файла bipolarrrz.m	29
2.31	Листинг файла manchester.m	29
2.32	Листинг файла diffmanc.m	29

2.33	Листинг файла calcspectre.m	30
2.34	Униполярное кодирование	31
2.35	Кодирование AMI	31
2.36	Кодирование NRZ	32
2.37	Кодирование RZ	32
2.38	Манчестерское кодирование	33
2.39	Дифференциальное манчестерское кодирование	33
2.40	Униполярное кодирование: нет самосинхронизации	34
2.41	Кодирование AMI: самосинхронизация при наличии сигнала	34
2.42	Кодирование NRZ: нет самосинхронизации	35
2.43	Кодирование RZ: есть самосинхронизация	35
2.44	Манчестерское кодирование: есть самосинхронизация	36
2.45	Дифференциальное манчестерское кодирование: есть са- мосинхронизация	36
2.46	Униполярное кодирование: спектр сигнала	37
2.47	Кодирование AMI: спектр сигнала	37
2.48	Кодирование NRZ: спектр сигнала	38
2.49	Кодирование RZ: спектр сигнала	38
2.50	Манчестерское кодирование: спектр сигнала	39
2.51	Дифференциальное манчестерское кодирование: спектр сигнала	39

Список таблиц

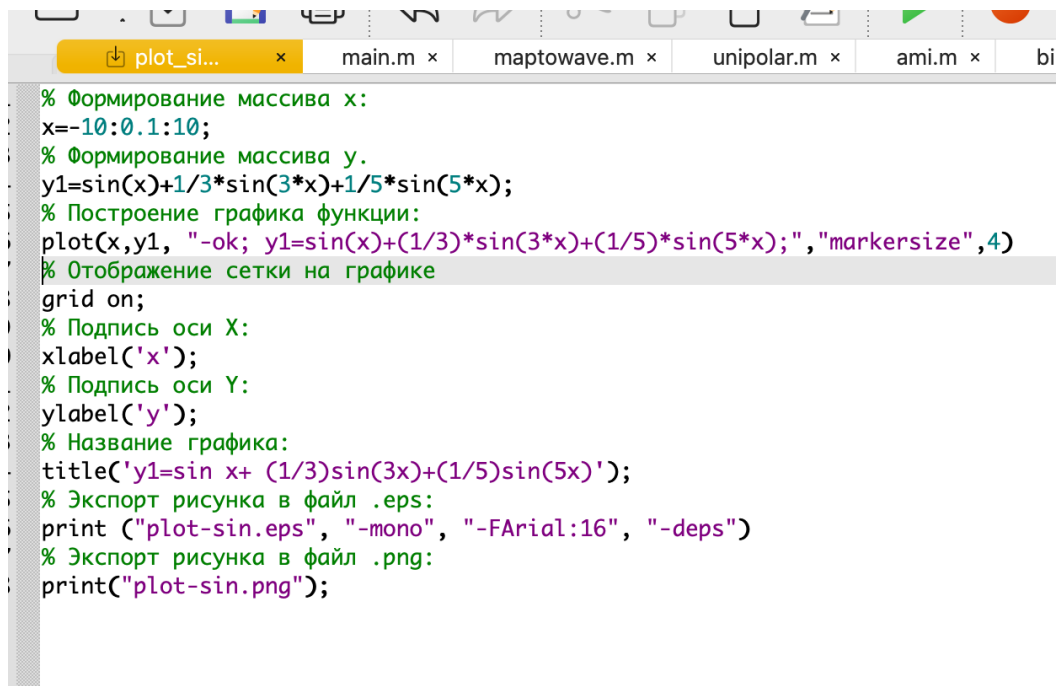
1 Цель работы

Целью данной работы является 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 Выполнение лабораторной работы

2.1 Построение графиков в Octave

Запускаем Octave, создаем новый сценарий под названием plot_sin.m. В окне редактора повторяем листинг по построению графика функции $y = \sin x + \frac{1}{3} \sin 3x + \frac{1}{5} \sin 5x$ на интервале $[-10; 10]$ (рис. 2.1).



```
% Формирование массива x:
x=-10:0.1:10;
% Формирование массива y.
y1=sin(x)+1/3*sin(3*x)+1/5*sin(5*x);
% Построение графика функции:
plot(x,y1, "-ok; y1=sin(x)+(1/3)*sin(3*x)+(1/5)*sin(5*x);","markersize",4)
% Отображение сетки на графике
grid on;
% Подпись оси X:
xlabel('x');
% Подпись оси Y:
ylabel('y');
% Название графика:
title('y1=sin x+ (1/3)sin(3x)+(1/5)sin(5x)');
% Экспорт рисунка в файл .eps:
print ("plot-sin.eps", "-mono", "-FArial:16", "-deps")
% Экспорт рисунка в файл .png:
print("plot-sin.png");
```

Рис. 2.1: Листинг файла plot_sin.m

Запускаем сценарий на выполнение, открывается окно с графиком (рис. 2.2). В рабочем каталоге появляются файлы с графиками в форматах .eps, .png (рис. 2.3).

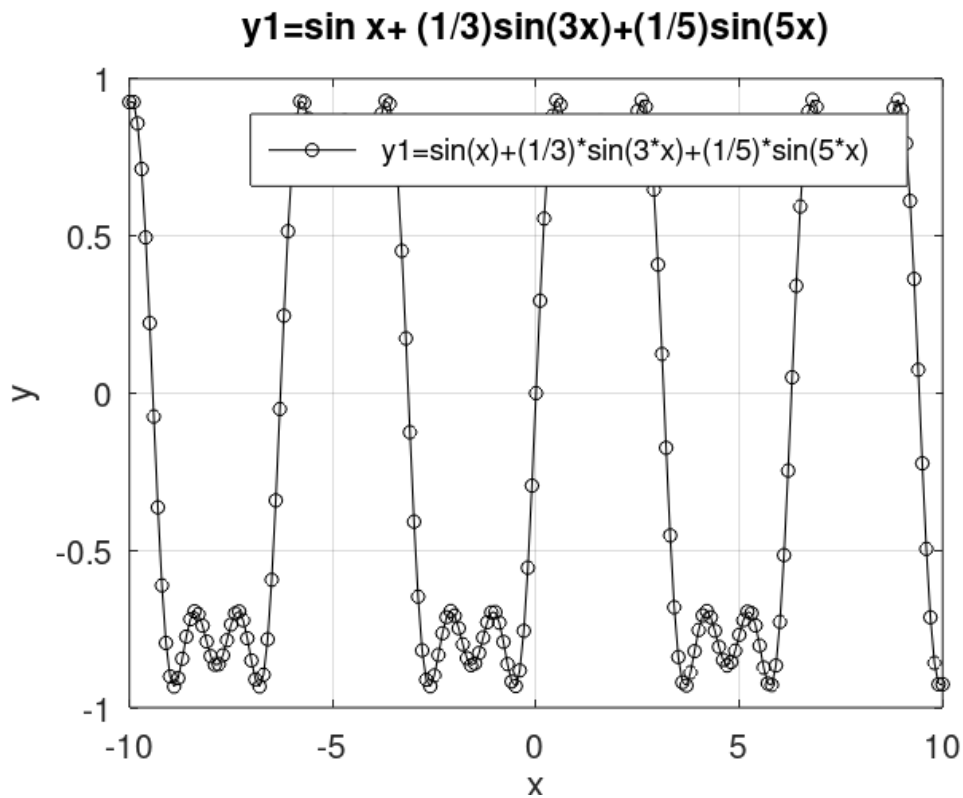


Рис. 2.2: График функций y_1 на интервале $-10; 10$

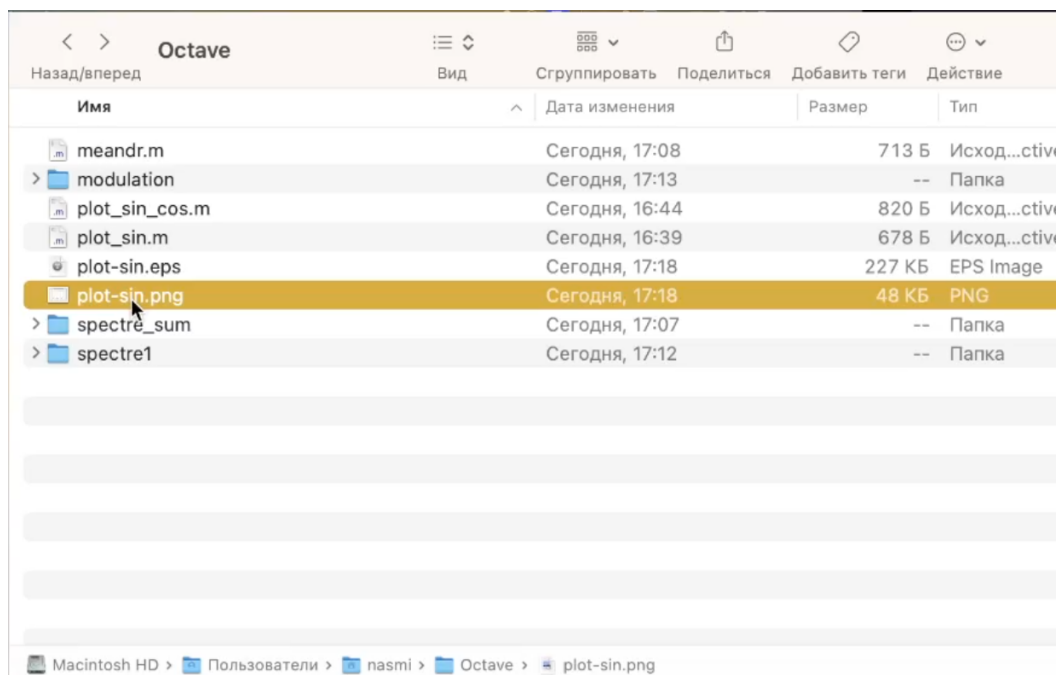


Рис. 2.3: Файлы .eps, .png

Сохраним сценарий под названием `plot_sin_cos.m` и изменим его так, чтобы на одном графике располагались отличающиеся по типу линий

графики функций $y_1 = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$, $y_2 = \cos x + 1/3 \cos 3x + 1/5 \cos 5x$. Итоговый листинг (рис. 2.4).

```

1  % Формирование массива x:
2  x=-10:0.1:10;
3  % Формирование массивов y1 и y2.
4  y1=sin(x)+1/3*sin(3*x)+1/5*sin(5*x);
5  y2=cos(x)+(1/3)*cos(3*x)+(1/5)*cos(5*x);
6  % Построение первого графика функции:
7  plot(x,y1, "-ok; y1=sin(x)+(1/3)*sin(3*x)+(1/5)*sin(5*x);", "markersize",4)
8  hold on
9  7
10
11 % Построение второго графика функции:
12 plot(x,y2, "-; y2=cos(x)+(1/3)*cos(3*x)+(1/5)*cos(5*x);", "markersize",4)
13 % Отображение сетки на графике
14 grid on;
15 % Подпись оси X:
16 xlabel('x');
17 % Подпись оси Y:
18 ylabel('y');
19 % Экспорт рисунка в файл .eps:
20 print ("plot-sin-cos.eps", "-mono", "-FArial:16", "-deps")
21 % Экспорт рисунка в файл .png:
22 print("plot-sin-cos.png");
23

```

Рис. 2.4: Листинг файла plot_sin_cos.m

Запускаем, получаем еще один график (рис. 2.5).

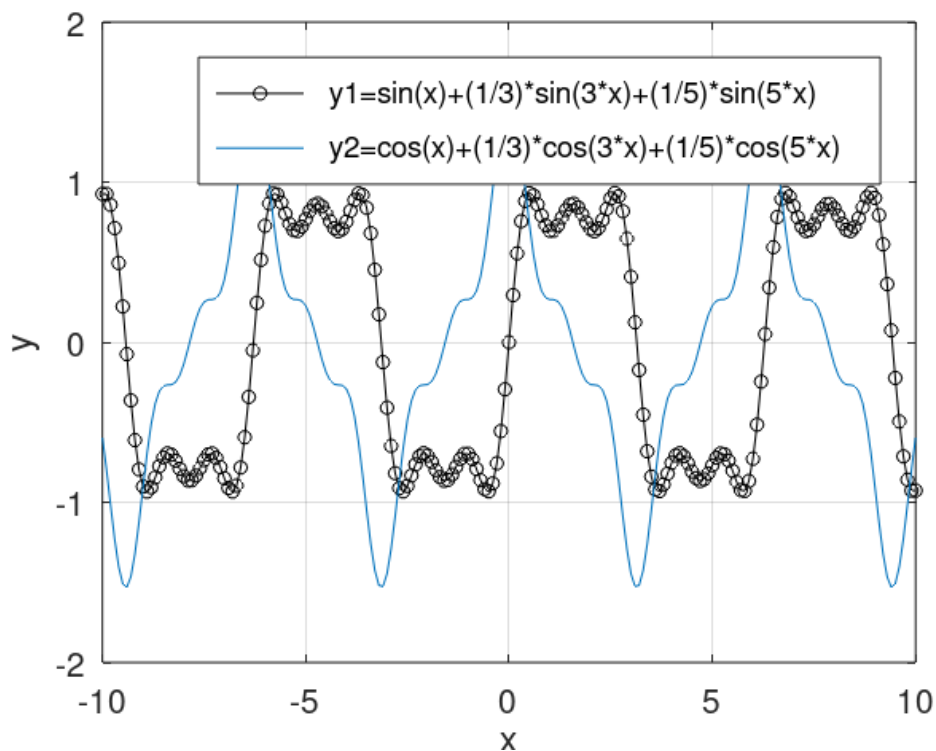


Рис. 2.5: График функций y_1 и y_2 на интервале $-10; 10$

2.2 Разложение импульсного сигнала в частичный ряд Фурье

Создадим новый сценарий `meandr.m`. В коде зададим начальные значения. Вычислим амплитуду гармоник и заполним массивы гармоник и элементов ряда. Далее задаём массив значений гармоник массив элементов ряда. 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. Также экспортируем полученный график в файл в формате `.png` (рис. 2.6), (рис. 2.7).

```
mean... x main.m x maptowave.m x
% meandr.m
% количество отсчетов (гармоник):
N=8;
% частота дискретизации:
t=-1:0.01:1;
% значение амплитуды:
A=1;
% период:
T=1;
% амплитуда гармоник
nh=(1:N)*2-1;
% массив коэффициентов для ряда, заданного через cos:
Am=2/pi ./ nh;
Am(2:2:end) = -Am(2:2:end);
% массив гармоник:
harmonics=cos(2 * pi * nh' * t/T);
% массив элементов ряда:
s1=harmonics.*repmat(Am',1,length(t));
% Суммирование ряда:
s2=cumsum(s1);
% Построение графиков:
for k=1:N
    subplot(4,2,k)
    plot(t, s2(k,:))
end
print("plot-meandr-cos.png");
```

Рис. 2.6: Листинг файла meandr.m

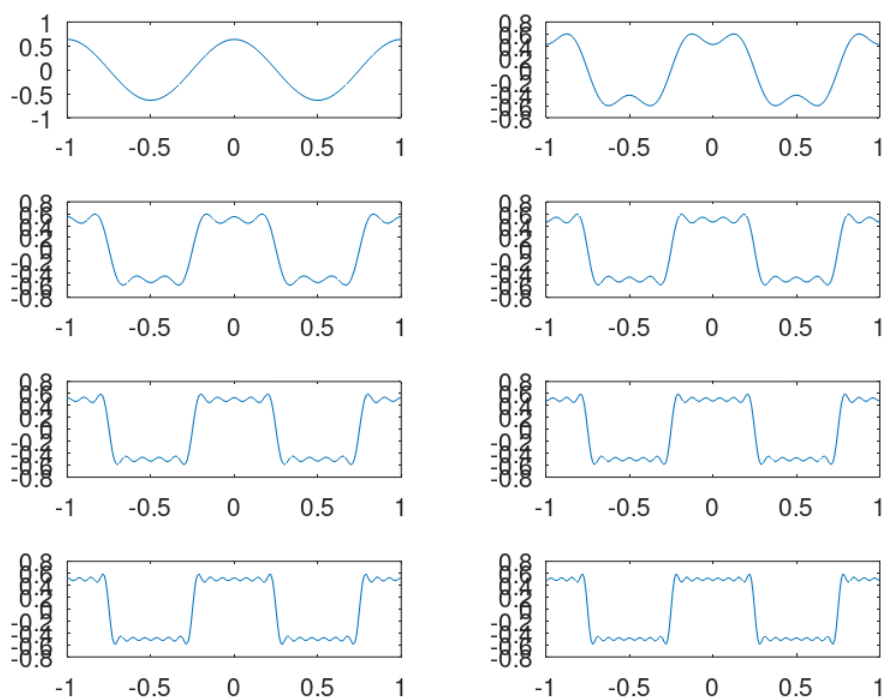


Рис. 2.7: Меандр через косинусы

Также реализуем меандр через синусы (рис. 2.8), (рис. 2.9).

```
mean... x main.m x maptowave.m x
% meandr.m
% количество отсчетов (гармоник):
N=8;
% частота дискретизации:
t=-1:0.01:1;
% значение амплитуды:
A=1;
% период:
T=1;
% амплитуда гармоник
nh=(1:N)*2-1;
% массив коэффициентов для ряда, заданного через sin:
Am=2/pi ./ nh;
Am(2:2:end) = -Am(2:2:end);
% массив гармоник:
harmonics=sin(2 * pi * nh' * t/T);
% массив элементов ряда:
s1=harmonics.*repmat(Am',1,length(t));
% Суммирование ряда:
s2=cumsum(s1);
% Построение графиков:
for k=1:N
    subplot(4,2,k)
    plot(t, s2(k,:))
end
print("plot-meandr-sin.png");
```

Рис. 2.8: Листинг файла meandr.m

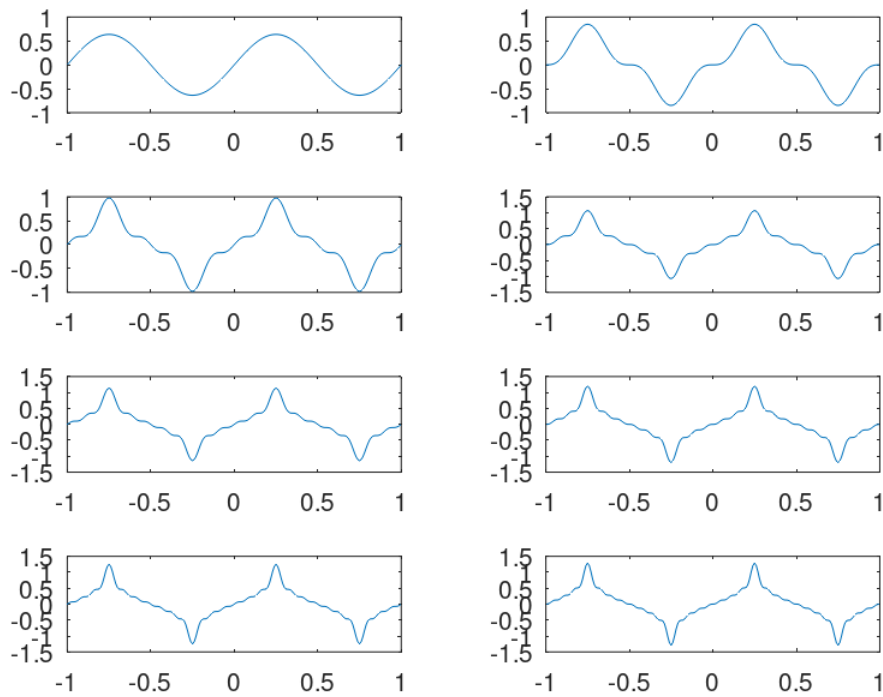


Рис. 2.9: Меандр через синусы

2.3 Определение спектра и параметров сигнала

Создадим в рабочем каталоге каталог `spectre1` и в нем новый сценарий `spectre.m`. В коде сценария зададим начальные значения, а также два синусоидальных сигнала разной частоты, построим графики сигналов (рис. 2.10), (рис. 2.11).

```

spectre.m ×   main.m ×   maptowave.m ×   unipola
% 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 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;
% Массив отсчётов 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. 2.10: Листинг файла spectre.m

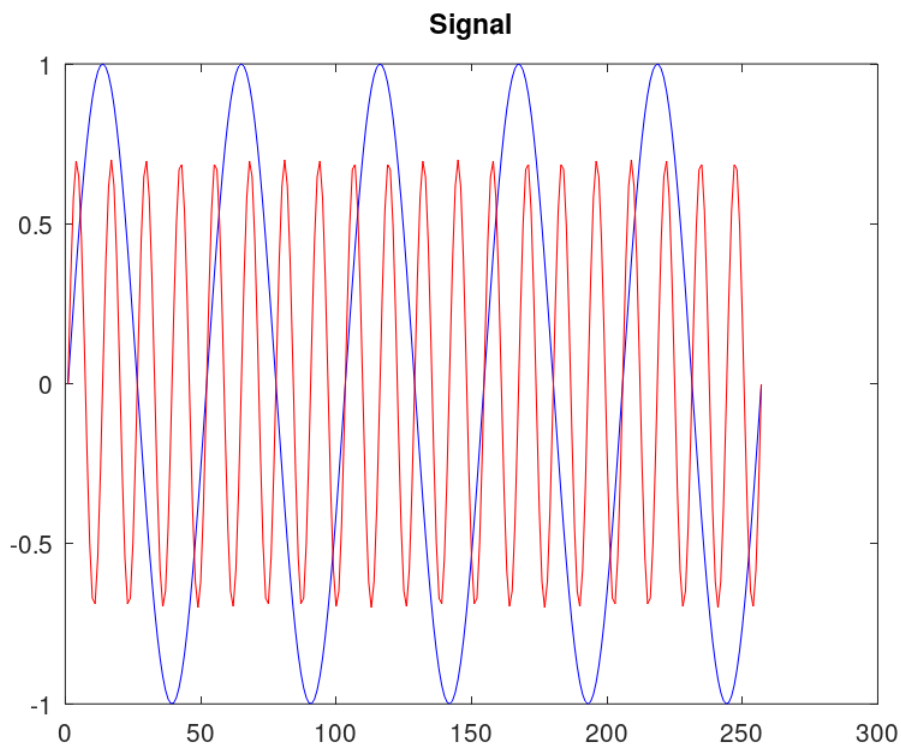


Рис. 2.11: Графики сигналов разной частоты

Затем с помощью быстрого преобразования Фурье найдем спектры сигналов, добавив в файл `spectre.m` код из мануала в ТУИСе. Учитывая реализацию преобразования Фурье, скорректируем график спектра (рис. 2.12): отбросим дублирующие отрицательные частоты, а также примем в расчёт то, что на каждом шаге вычисления быстрого преобразования Фурье происходит суммирование амплитуд сигналов.


```

% 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. 2.12: Листинг файла spectre.m

Получим следующие графики: график спектров синусоидальных сигналов (рис. 2.13) и исправленный график спектров синусоидальных сигналов (рис. 2.14).

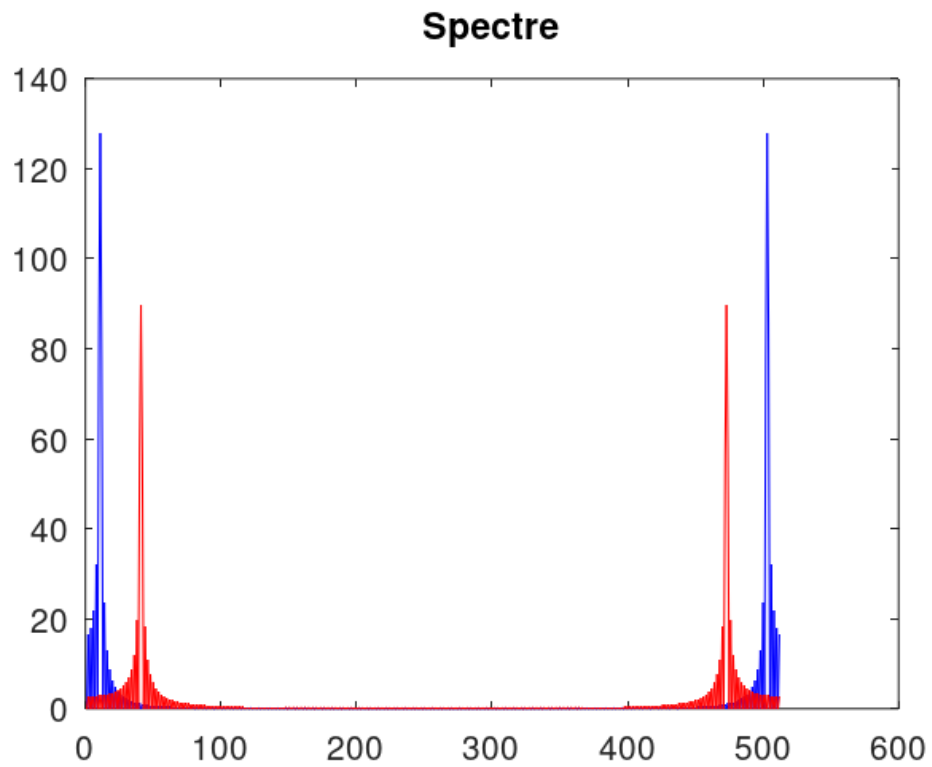


Рис. 2.13: График спектра синусоидальных сигналов

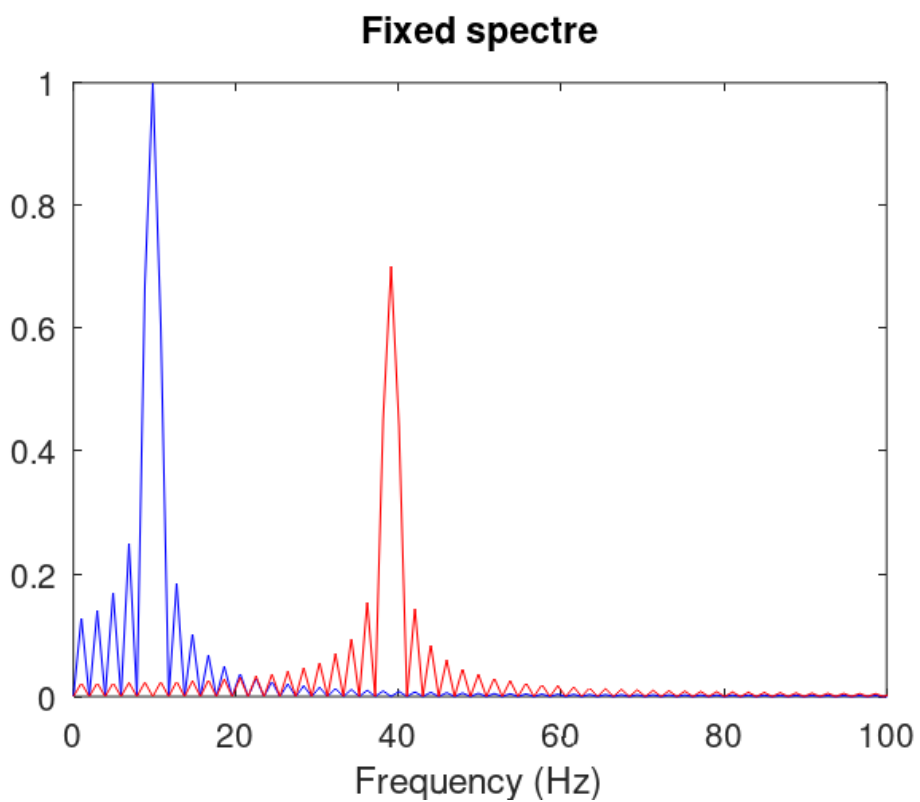
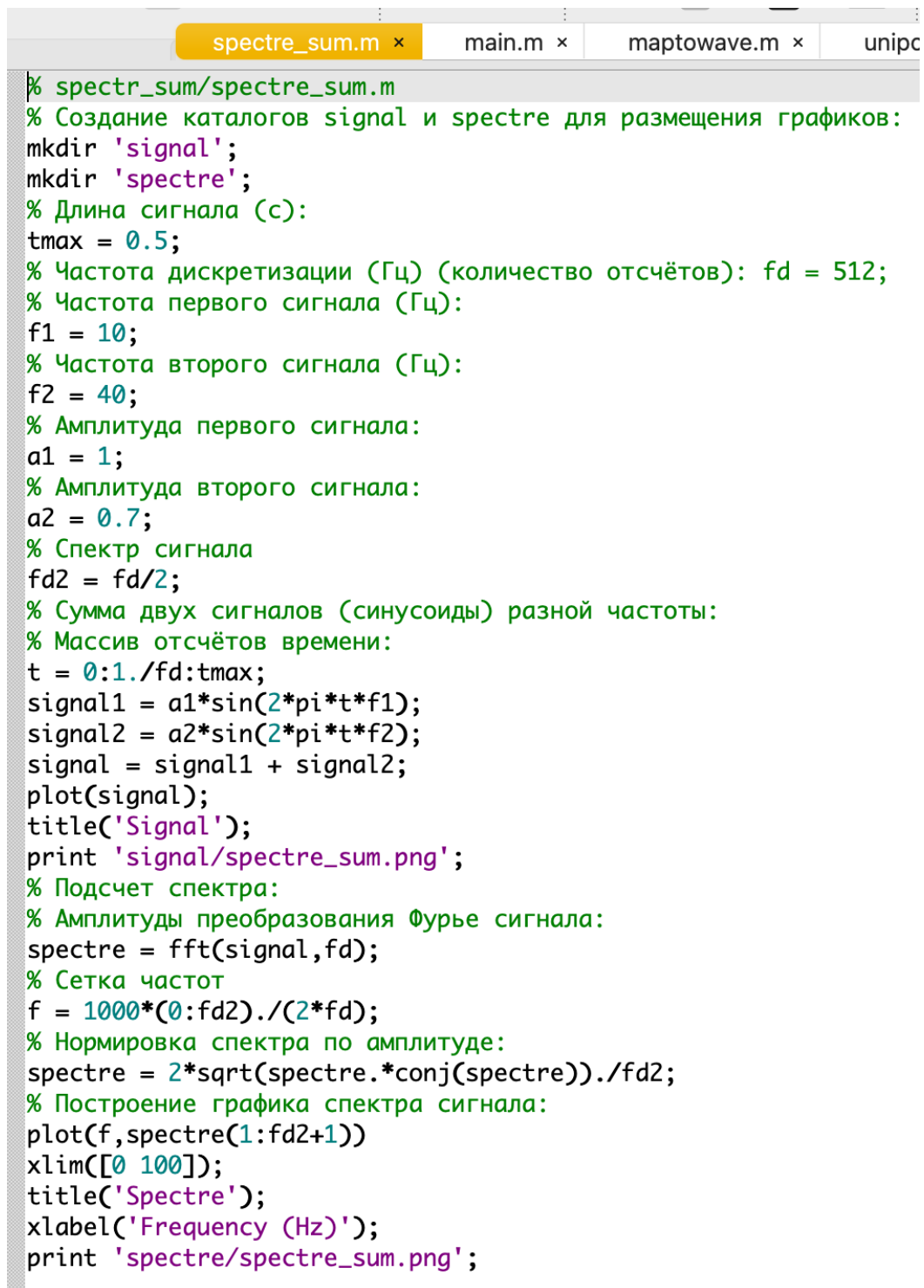


Рис. 2.14: Исправленный график спектров синусоидальных сигналов

Найдем спектр суммы рассмотренных сигналов, создадим каталог

spectr_sum и в нем spectre_sum.m (рис. 2.15), (рис. 2.16).



```
% spectr_sum/spectre_sum.m
% Создание каталогов signal и spectre для размещения графиков:
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;
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. 2.15: Листинг файла spectre_sum.m

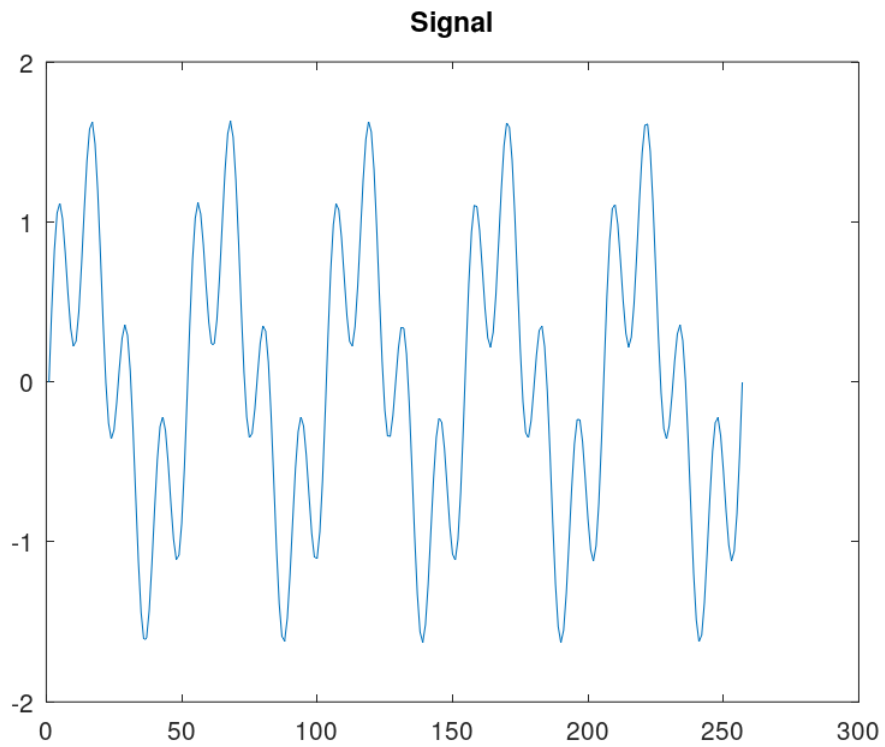


Рис. 2.16: Суммарный сигнал

В результате должен получиться аналогичный предыдущему результат (рис. 2.17), т.е. спектр суммы сигналов должен быть равен сумме спектров сигналов, что вытекает из свойств преобразования Фурье.

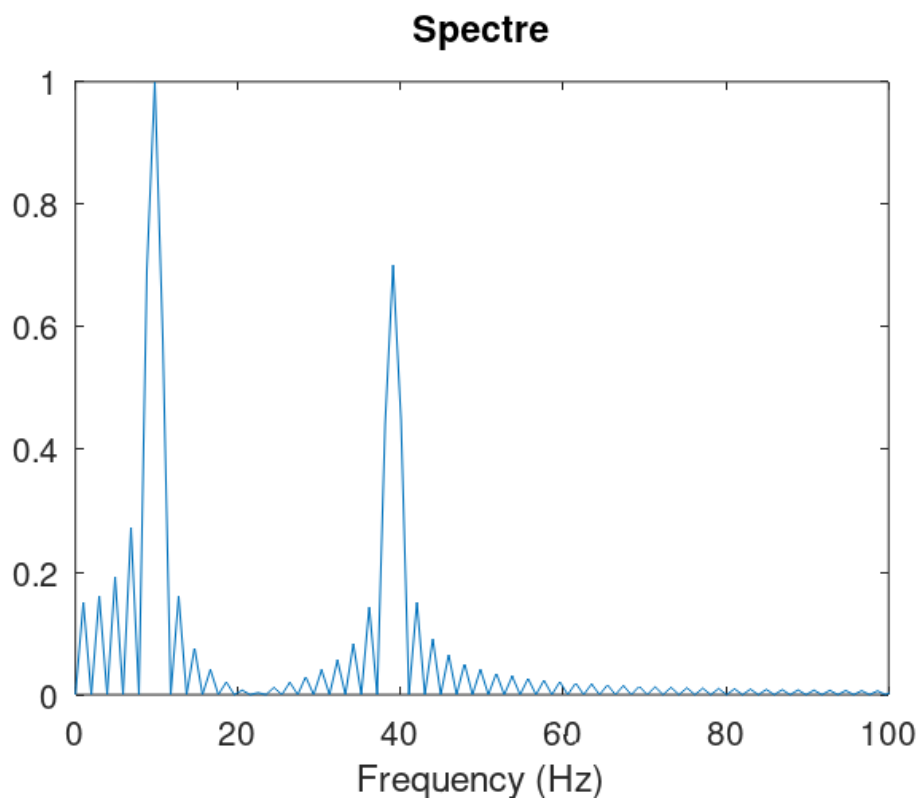


Рис. 2.17: Спектр суммарного сигнала

2.4 Амплитудная модуляция

В рабочем каталоге создадим каталог `modulation` и в нём новый сценарий с именем `am.m`. Добавим в него код из мануала (рис. 2.18).

```

am.m ×    main.m ×    maptowave.m ×    unipolar.r
% modulation/am.m
% Создание каталогов signal и spectre для размещения 4 графиков:
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;
plot(signal, 'b');
hold on
% Построение огибающей:
plot(signal1, 'r');
plot(-signal1, 'r');
hold off
title('Signal');
print 'signal/am.png';
% 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:
plot(f,spectre(1:fd2+1), 'b')
xlim([0 100]);
title('Spectre');
xlabel('Frequency (Hz)');
print 'spectre/am.png';

```

Рис. 2.18: Листинг файла am.m

В результате получаем, что спектр произведения представляет собой свертку спектров (рис. 2.19), (рис. 2.20).

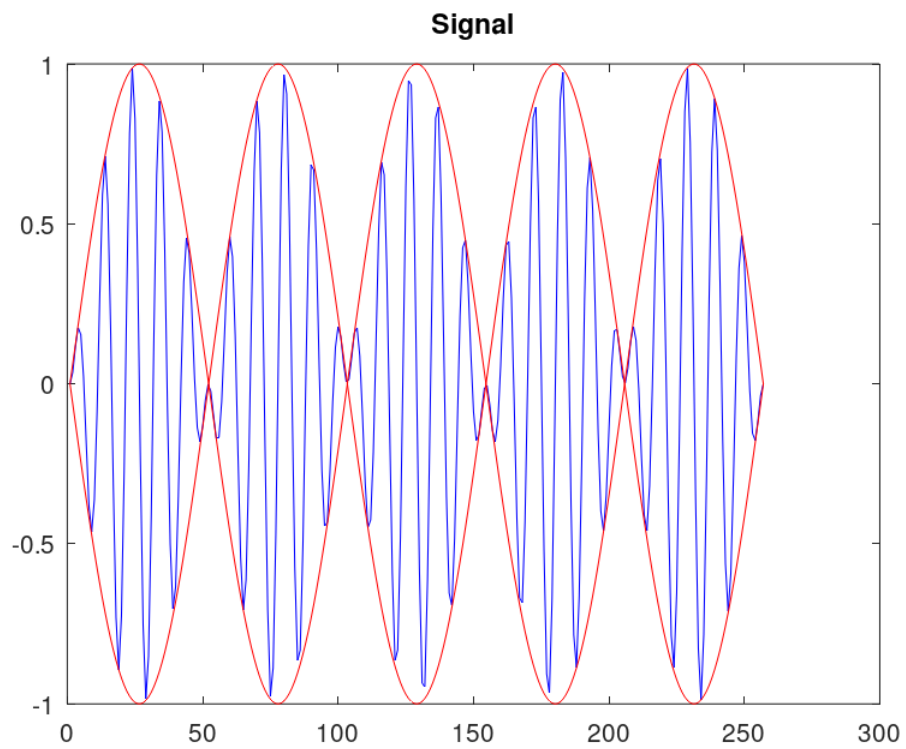


Рис. 2.19: Сигнал и огибающая при амплитудной модуляции

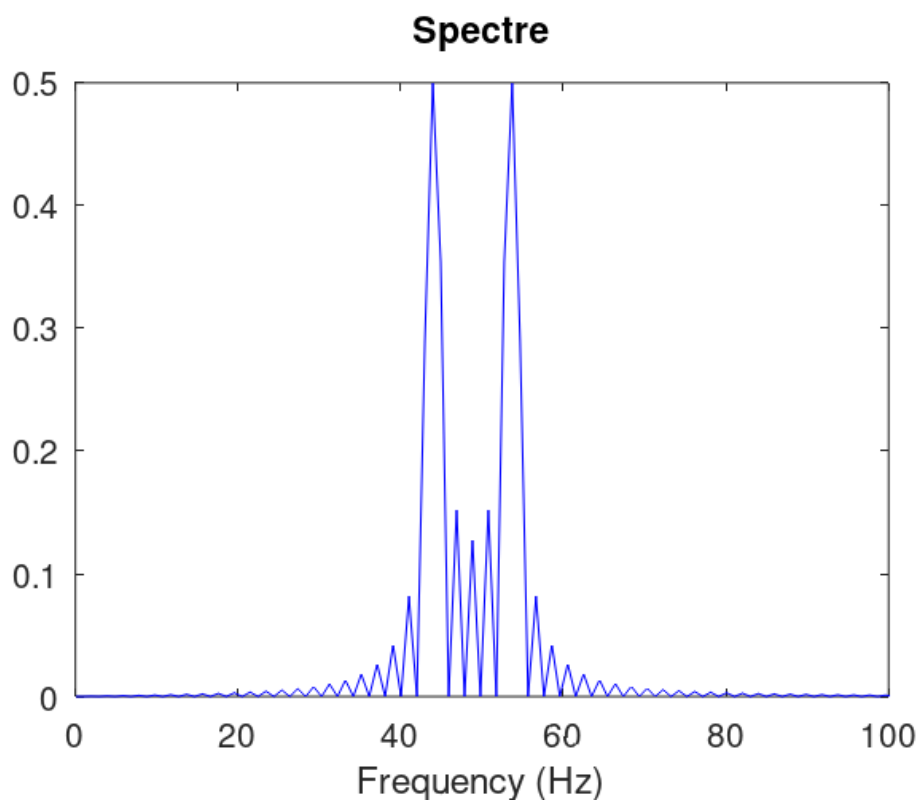


Рис. 2.20: Спектр сигнала при амплитудной модуляции

2.5 Кодирование сигнала. Исследование свойства самосинхронизации сигнала

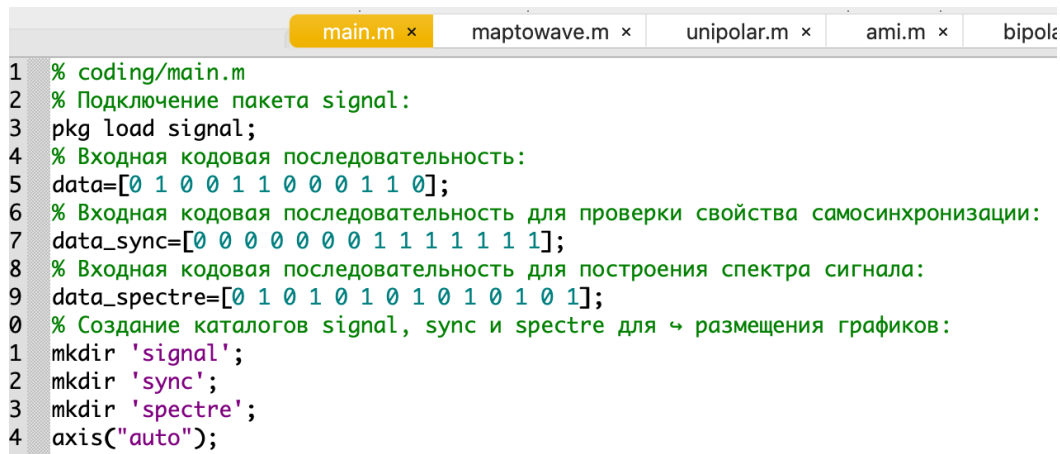
В рабочем каталоге создадим каталог coding и в нём файлы main.m, maptowave.m, unipolar.m, ami.m, bipolarnrz.m, bipolarrrz.m, manchester.m, diffmanc.m, calcspectre.m.

В окне интерпретатора команд проверяем, установлен ли пакет расширений signal: pkg list. Так как он не установлен, то устанавливаем его: pkg list -forge и pkg install control signal (рис. 2.21).

```
>> pkg list
Package Name | Version | Installation directory
-----+-----+-----
control      | 4.0.1   | /Users/nasmi/.local/share/octave/api-v59/packages/control-4.0.1
signal *| 1.4.5   | /Users/nasmi/.local/share/octave/api-v59/packages/signal-1.4.5
>> main
```

Рис. 2.21: Проверка правильности установки пакета signal

В файле main.m подключаем пакет signal и задаем входные кодовые последовательности (рис. 2.22).



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

Рис. 2.22: Задаем входные кодовые последовательности

Затем в этом же файле пропишем вызовы функций для построения графиков модуляций кодированных сигналов для кодовой последовательности data (рис. 2.23).


```

% Униполярное кодирование
wave=unipolar(data);
plot(wave);
ylim([-1 6]);
title('Unipolar');
print 'signal/unipolar.png';
% Кодирование ami
wave=ami(data);
plot(wave)
title('AMI');
print 'signal/ami.png';
% Кодирование NRZ
wave=bipolarnrz(data);
plot(wave);
title('Bipolar Non-Return to Zero');
print 'signal/bipolarnrz.png';
% Кодирование RZ
wave=bipolarrz(data);
plot(wave)
title('Bipolar Return to Zero');
print 'signal/bipolarrz.png';
% Манчестерское кодирование
wave=manchester(data);
plot(wave)
title('Manchester');
print 'signal/manchester.png';
% Дифференциальное манчестерское кодирование
wave=diffmanc(data);
plot(wave)
title('Differential Manchester');
print 'signal/diffmanc.png';

```

Рис. 2.23: Вызовы функций для построения модуляций кодированных сигналов кодовой последовательности data

Пропишем вызовы функций для построения графиков модуляций кодированных сигналов для кодовой последовательности data_sync (рис.

2.24).

```
% Униполярное кодирование
wave=unipolar(data_sync);
plot(wave);
ylim([-1 6]);
title('Unipolar');
print 'sync/unipolar.png';
% Кодирование AMI
wave=ami(data_sync);
plot(wave)
title('AMI');
print 'sync/ami.png';
% Кодирование NRZ
wave=bipolarnrz(data_sync);
plot(wave);
title('Bipolar Non-Return to Zero');
print 'sync/bipolarnrz.png';
% Кодирование RZ
wave=bipolarrz(data_sync);
plot(wave)
title('Bipolar Return to Zero');
print 'sync/bipolarrz.png';
% Манчестерское кодирование
wave=manchester(data_sync);
plot(wave)
title('Manchester');
print 'sync/manchester.png';
% Дифференциальное манчестерское кодирование
wave=diffmanc(data_sync);
plot(wave)
title('Differential Manchester');
print 'sync/diffmanc.png';
```

Рис. 2.24: Вызовы функций для построения модуляций кодированных сигналов кодовой последовательности data_sync

Далее в этом же файле пропишем вызовы функций для построения

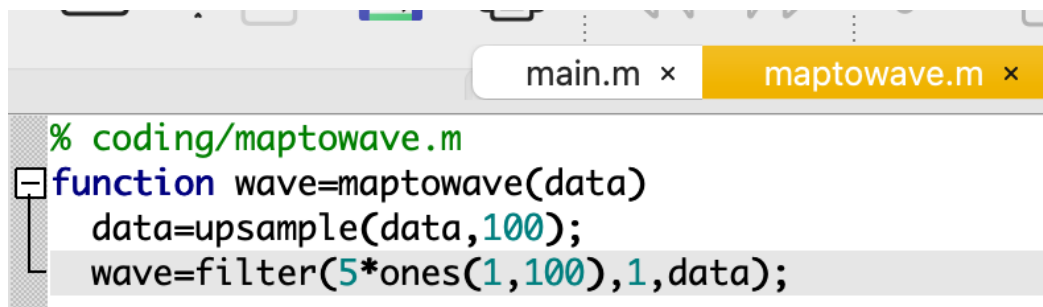
графиков спектров (рис. 2.25).

```
% Униполярное кодирование:
wave=unipolar(data_spectre);
spectre=calcspectre(wave);
title('Unipolar');
print 'spectre/unipolar.png';
% Кодирование AMI:
wave=ami(data_spectre);
spectre=calcspectre(wave);
title('AMI');
print 'spectre/ami.png';
% Кодирование NRZ:
wave=bipolarnrz(data_spectre);
spectre=calcspectre(wave);
title('Bipolar Non-Return to Zero');
print 'spectre/bipolarnrz.png';
% Кодирование RZ:
wave=bipolarrz(data_spectre);
spectre=calcspectre(wave);
title('Bipolar Return to Zero');
print 'spectre/bipolarrz.png';
% Манчестерское кодирование:
wave=manchester(data_spectre);
spectre=calcspectre(wave);
title('Manchester');
print 'spectre/manchester.png';
% Дифференциальное манчестерское кодирование:
wave=diffmanc(data_spectre);
spectre=calcspectre(wave);
title('Differential Manchester');
print 'spectre/diffmanc.png';
```

Рис. 2.25: Вызовы функций для построения графиков спектров

В файле `maptowave.m` пропишем функцию, которая по входному бито-

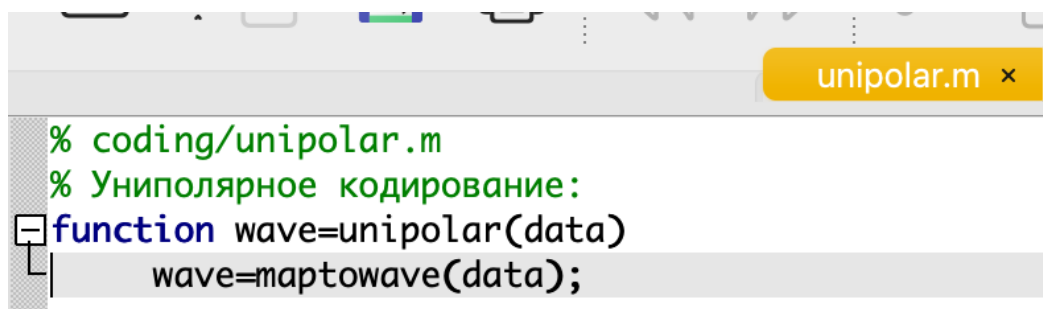
вому потоку строит график сигнала (рис. 2.26).

The image shows a MATLAB code editor window with two tabs: 'main.m' and 'maptowave.m'. The 'maptowave.m' tab is active. The code in the editor is as follows:

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

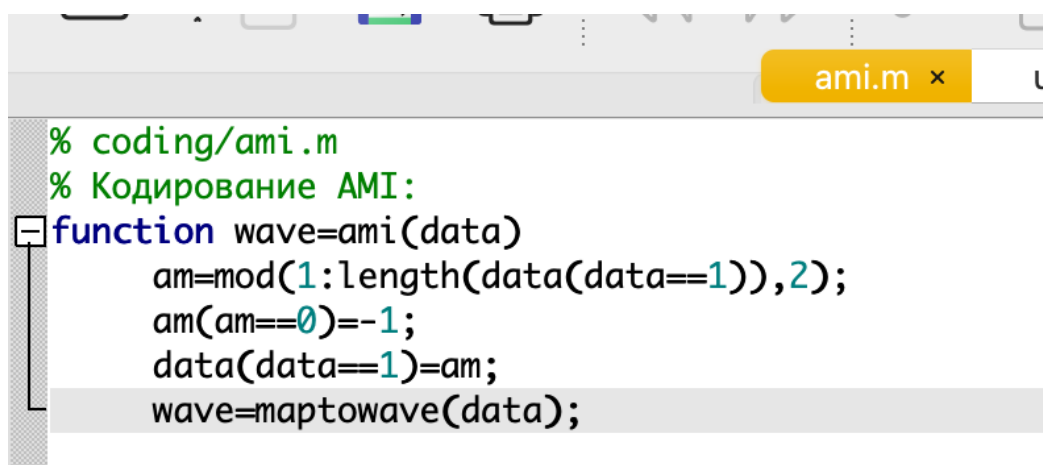
Рис. 2.26: Листинг файла maptowave.m

В файлах unipolar.m (рис. 2.27), ami.m (рис. 2.28), bipolarnrz.m (рис. 2.29), bipolarrrz.m (рис. 2.30), manchester.m (рис. 2.31), diffmanc.m (рис. 2.32) пропишем соответствующие функции преобразования кодовой последовательности data с вызовом функции maptowave для построения соответствующего графика.

The image shows a MATLAB code editor window with a tab for 'unipolar.m'. The code in the editor is as follows:

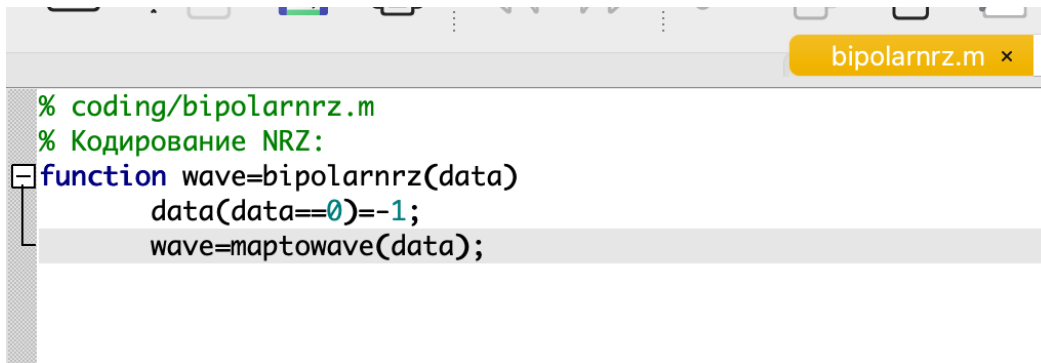
```
% coding/unipolar.m
% Униполярное кодирование:
function wave=unipolar(data)
    wave=maptowave(data);
```

Рис. 2.27: Листинг файла unipolar.m

The image shows a MATLAB code editor window with a tab for 'ami.m'. The code in the editor is as follows:

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

Рис. 2.28: Листинг файла ami.m

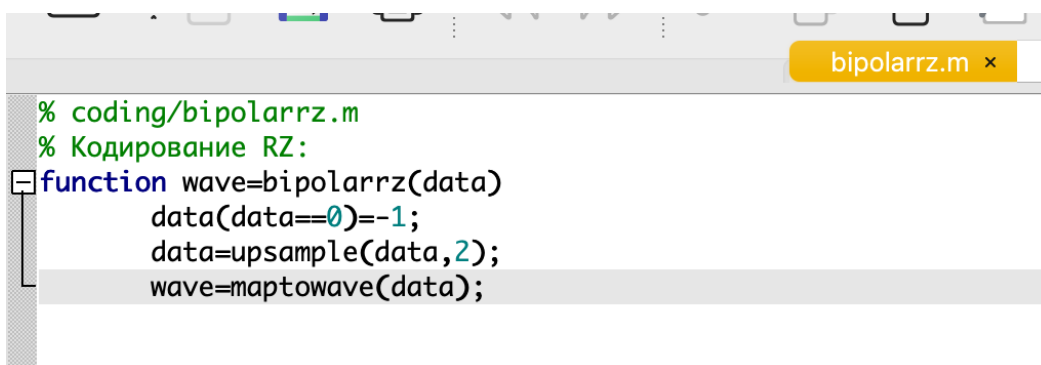


```

bipolarnrz.m x
% coding/bipolarnrz.m
% Кодирование NRZ:
function wave=bipolarnrz(data)
    data(data==0)=-1;
    wave=maptowave(data);

```

Рис. 2.29: Листинг файла bipolarnrz.m

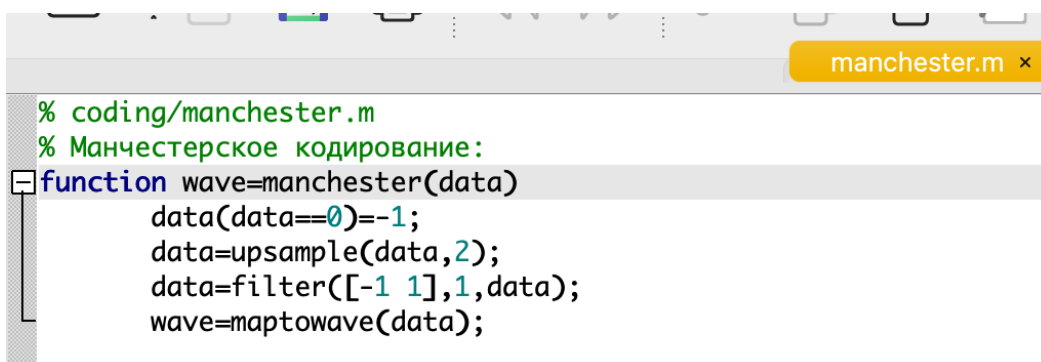


```

bipolarrrz.m x
% coding/bipolarrrz.m
% Кодирование RZ:
function wave=bipolarrrz(data)
    data(data==0)=-1;
    data=upsample(data,2);
    wave=maptowave(data);

```

Рис. 2.30: Листинг файла bipolarrrz.m

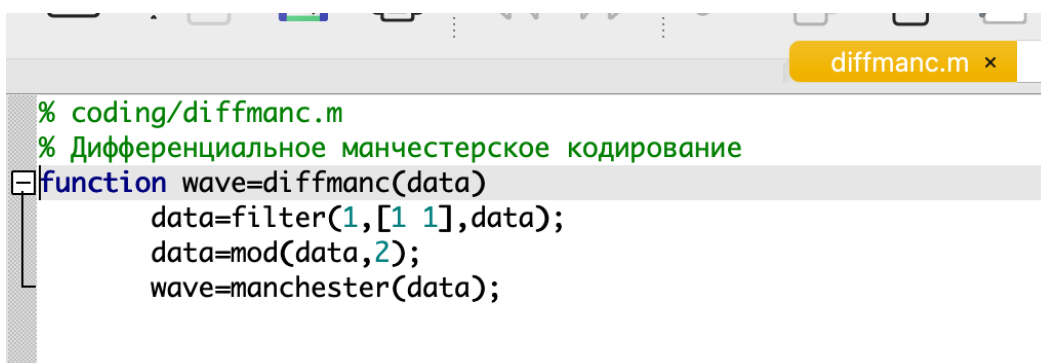


```

manchester.m x
% coding/manchester.m
% Манчестерское кодирование:
function wave=manchester(data)
    data(data==0)=-1;
    data=upsample(data,2);
    data=filter([-1 1],1,data);
    wave=maptowave(data);

```

Рис. 2.31: Листинг файла manchester.m



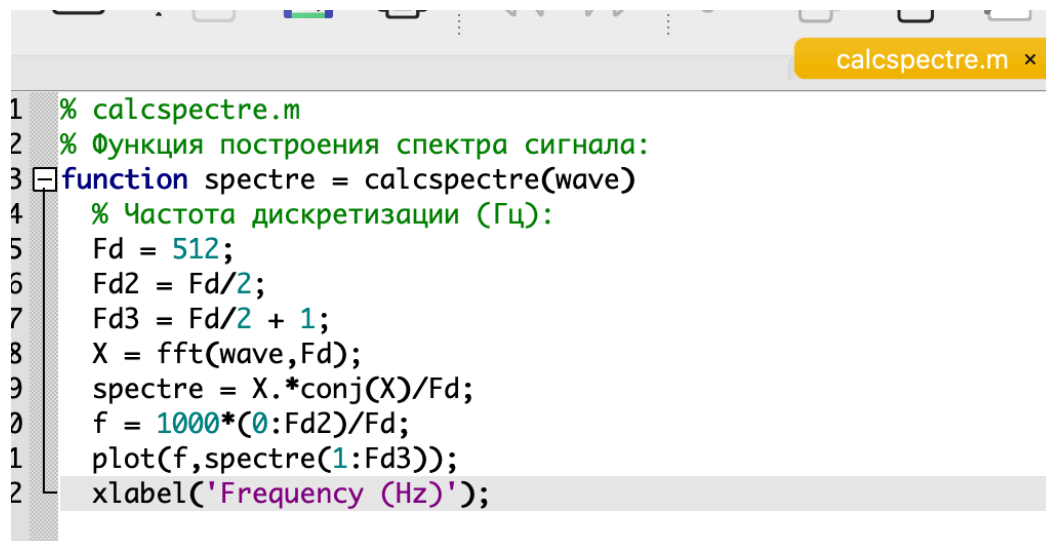
```

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

```

Рис. 2.32: Листинг файла diffmanc.m

В файле `calcspectre.m` пропишем функцию построения спектра сигнала (рис. 2.33).



```
1 % calcspectre.m
2 % Функция построения спектра сигнала:
3 function spectre = calcspectre(wave)
4 % Частота дискретизации (Гц):
5 Fd = 512;
6 Fd2 = Fd/2;
7 Fd3 = Fd/2 + 1;
8 X = fft(wave,Fd);
9 spectre = X.*conj(X)/Fd;
0 f = 1000*(0:Fd2)/Fd;
1 plot(f,spectre(1:Fd3));
2 xlabel('Frequency (Hz)');
```

Рис. 2.33: Листинг файла `calcspectre.m`

Запустим главный скрипт `main.m`. В каталоге `signal` должны быть получены файлы с графиками кодированного сигнала (рис. 2.34-2.39), в каталоге `sync` — файлы с графиками, иллюстрирующими свойства самосинхронизации (рис. 2.40-2.45), в каталоге `spectre` — файлы с графиками спектров сигналов (рис. 2.46-2.51).

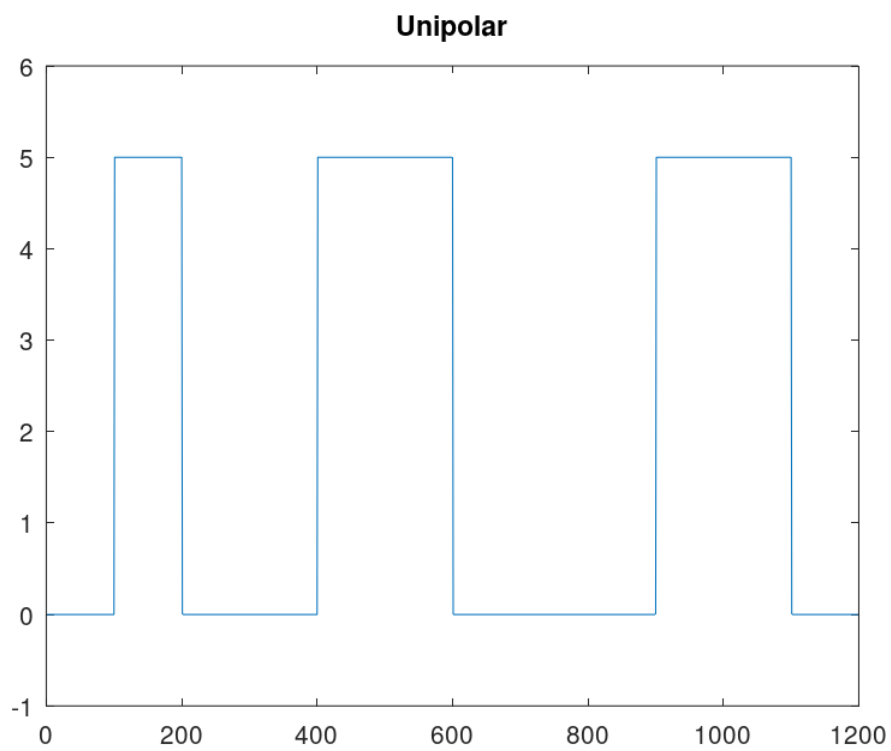


Рис. 2.34: Униполярное кодирование

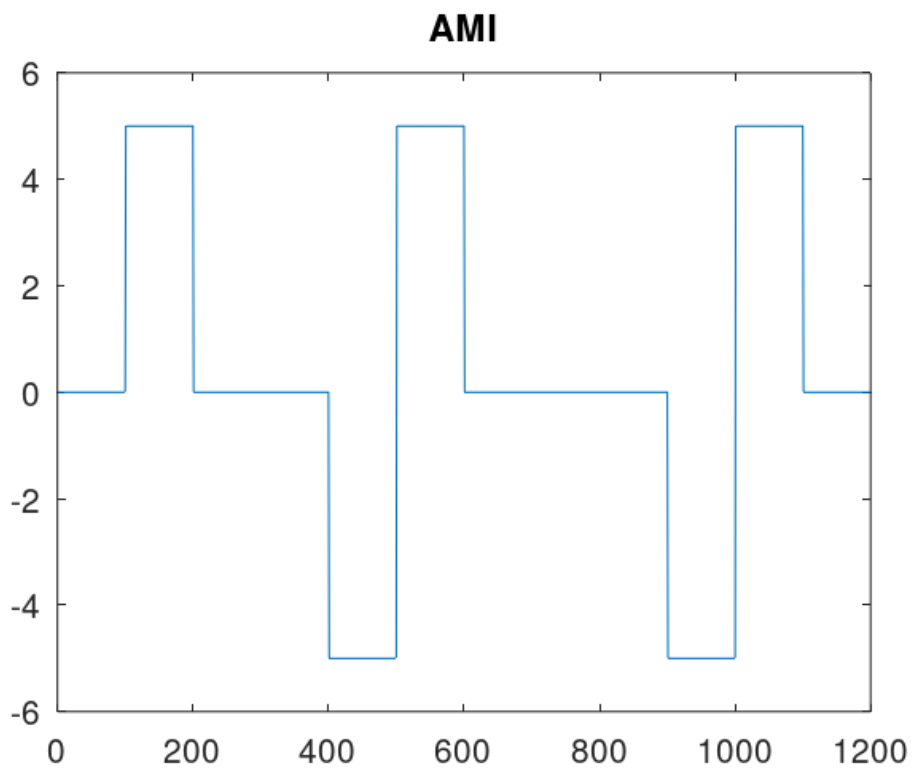


Рис. 2.35: Кодирование AMI

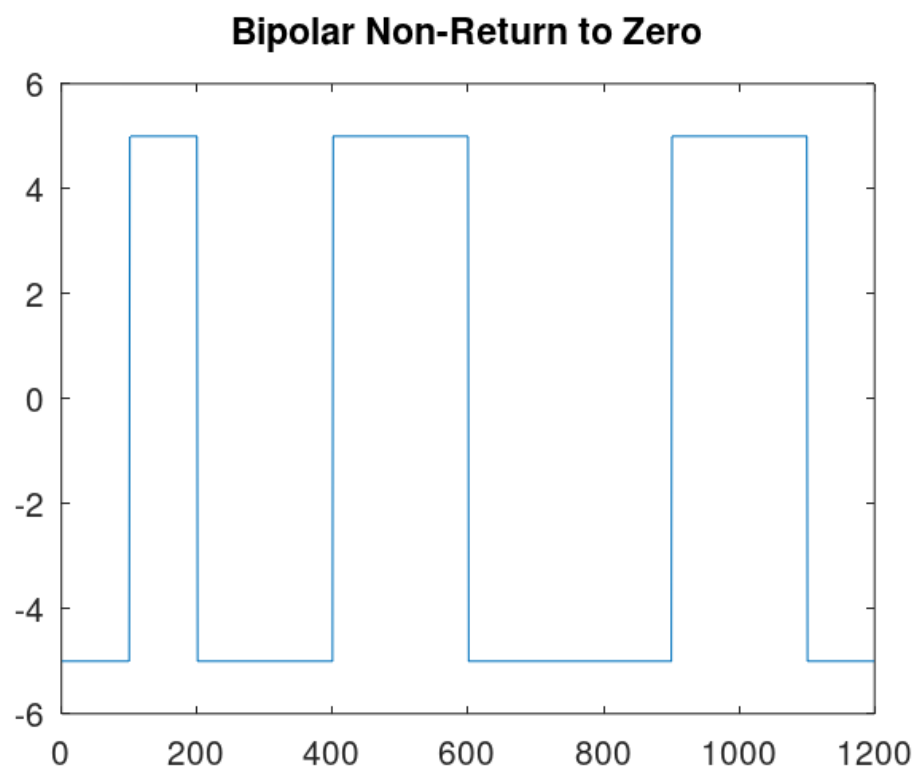


Рис. 2.36: Кодирование NRZ

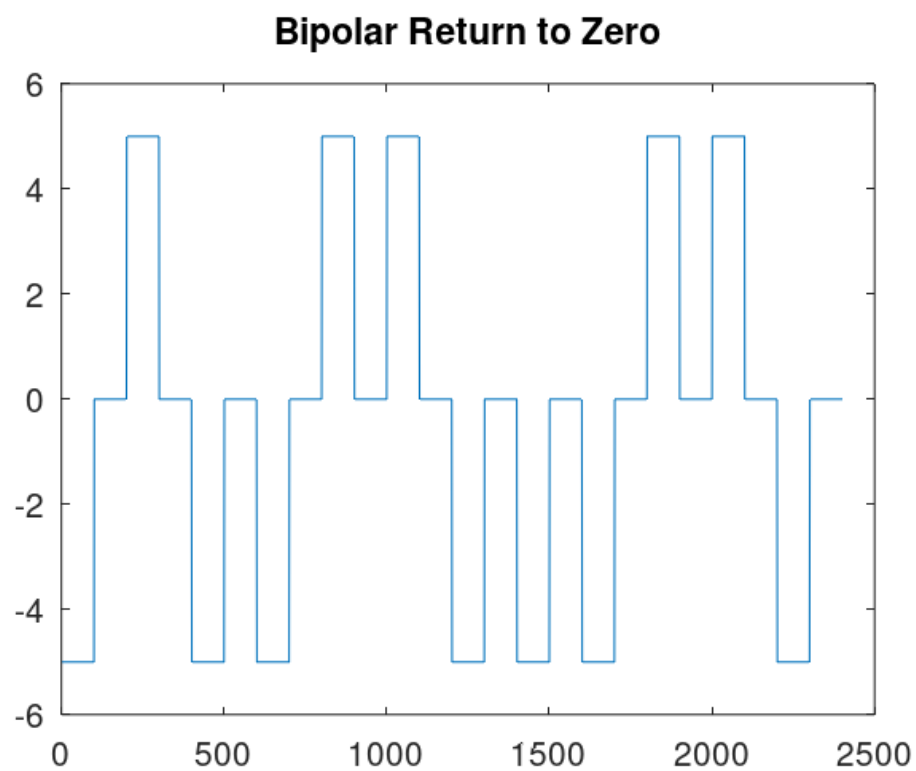


Рис. 2.37: Кодирование RZ

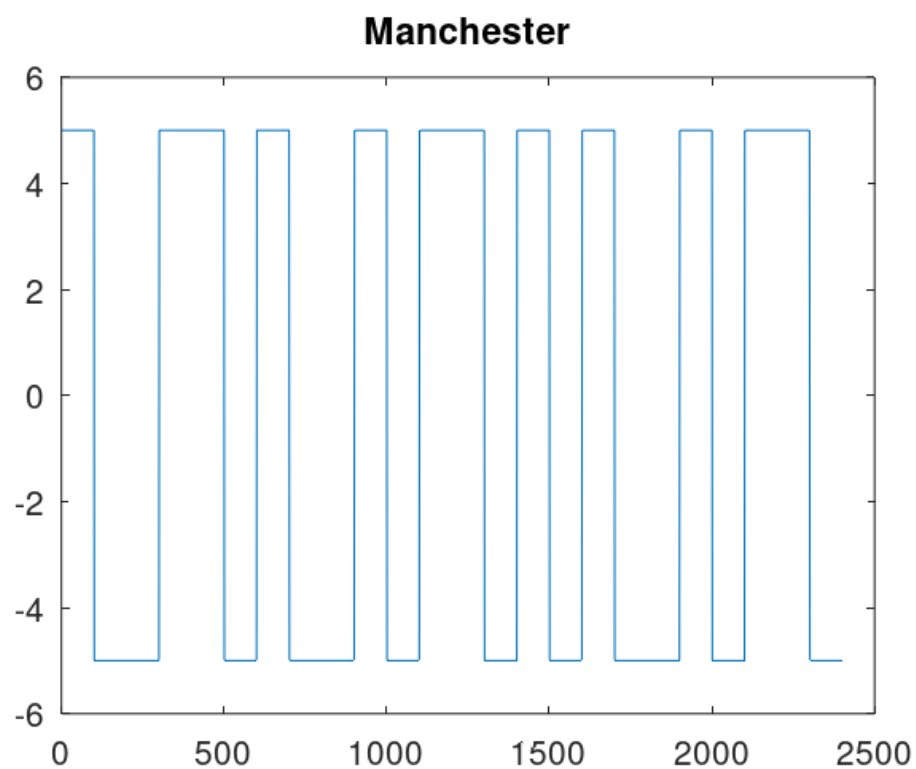


Рис. 2.38: Манчестерское кодирование

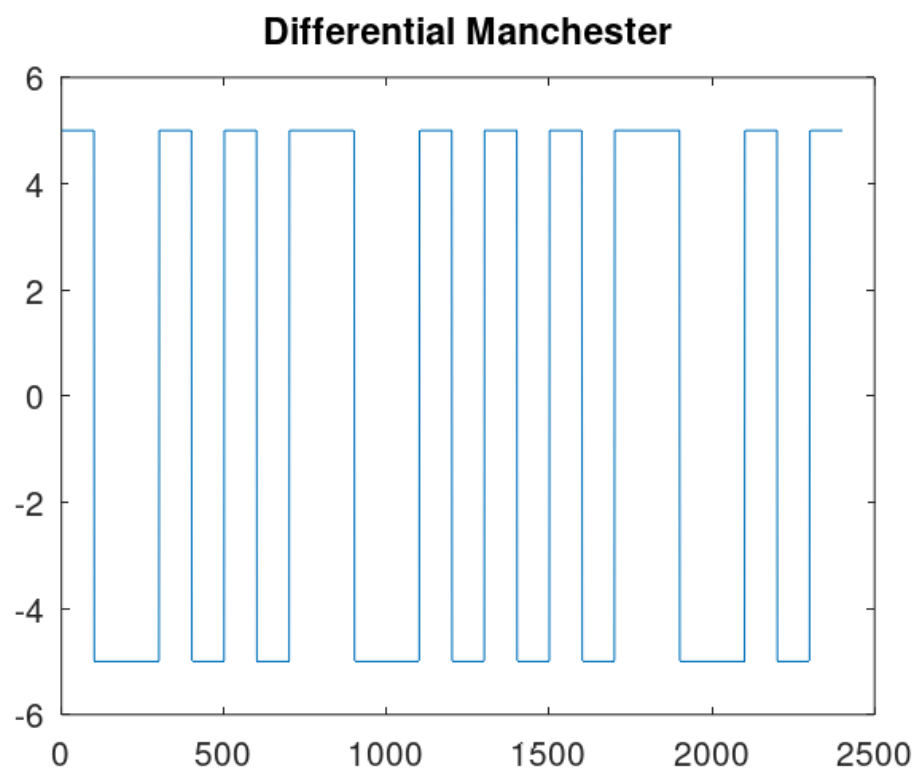


Рис. 2.39: Дифференциальное манчестерское кодирование

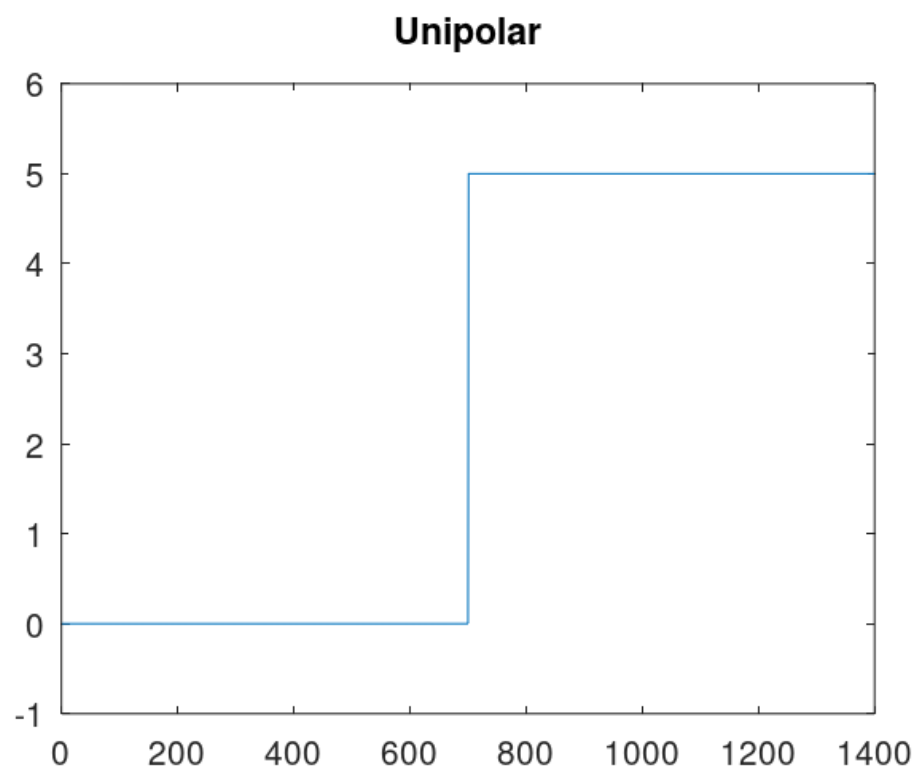


Рис. 2.40: Униполярное кодирование: нет самосинхронизации

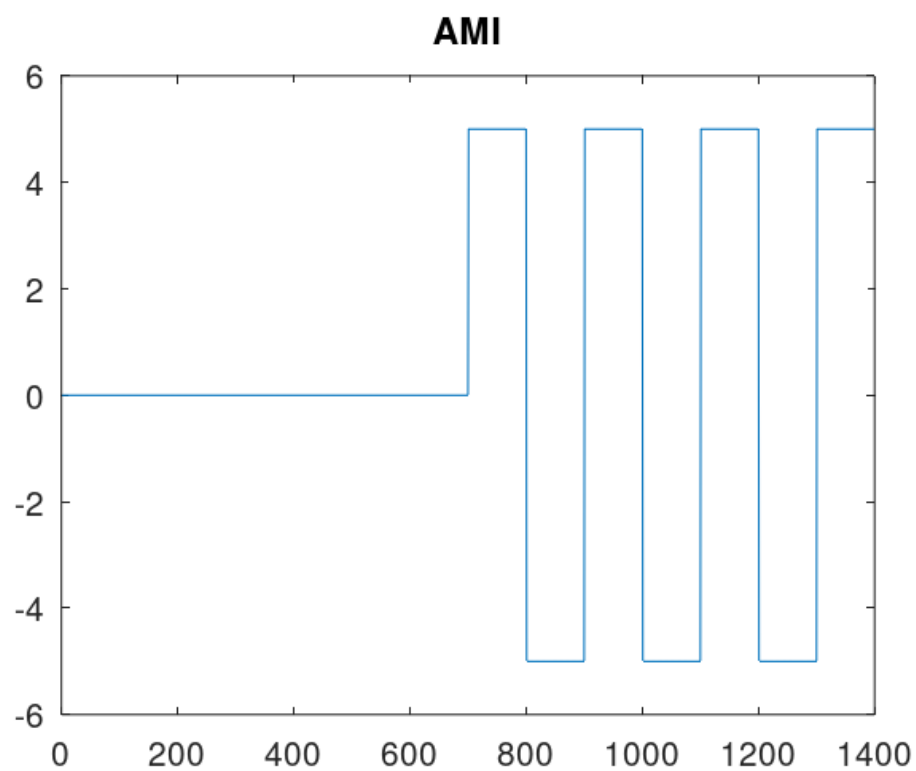


Рис. 2.41: Кодирование AMI: самосинхронизация при наличии сигнала

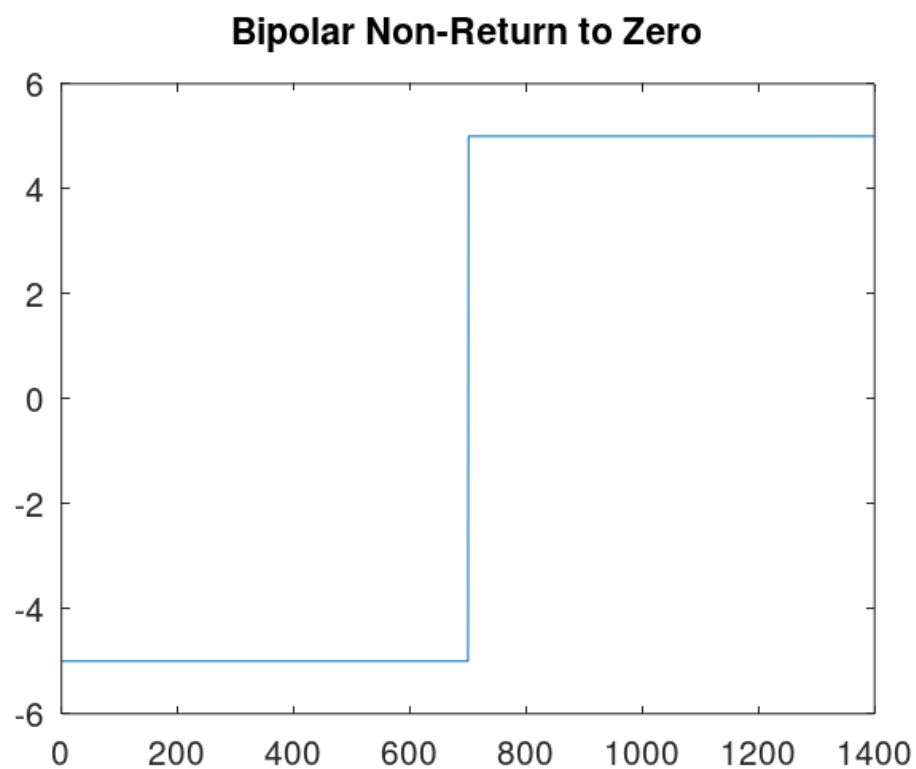


Рис. 2.42: Кодирование NRZ: нет самосинхронизации

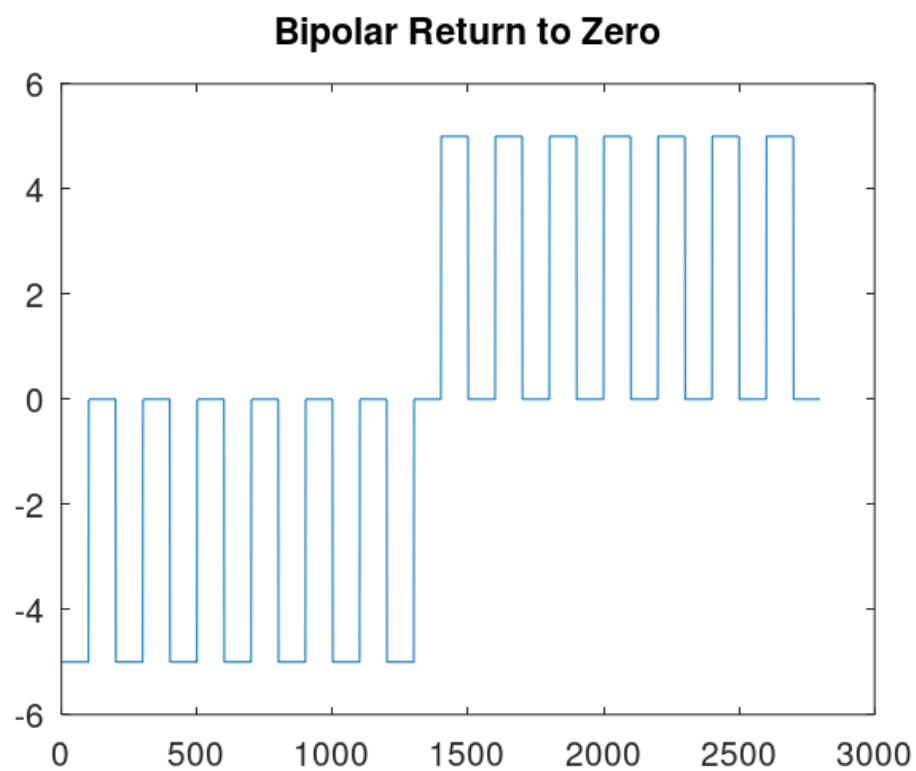


Рис. 2.43: Кодирование RZ: есть самосинхронизация

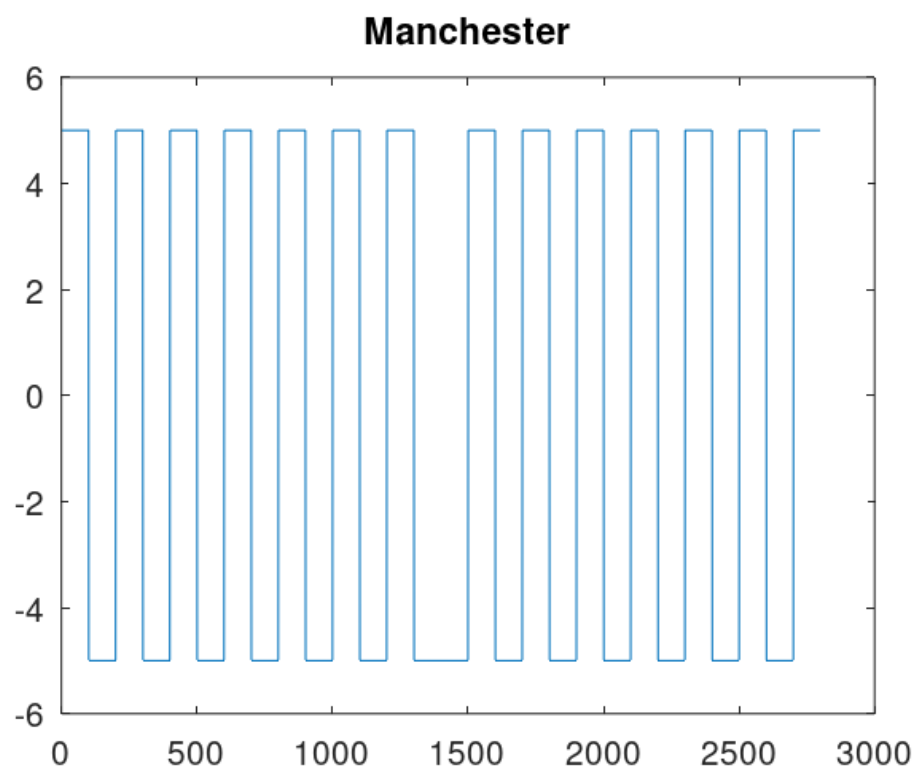


Рис. 2.44: Манчестерское кодирование: есть самосинхронизация

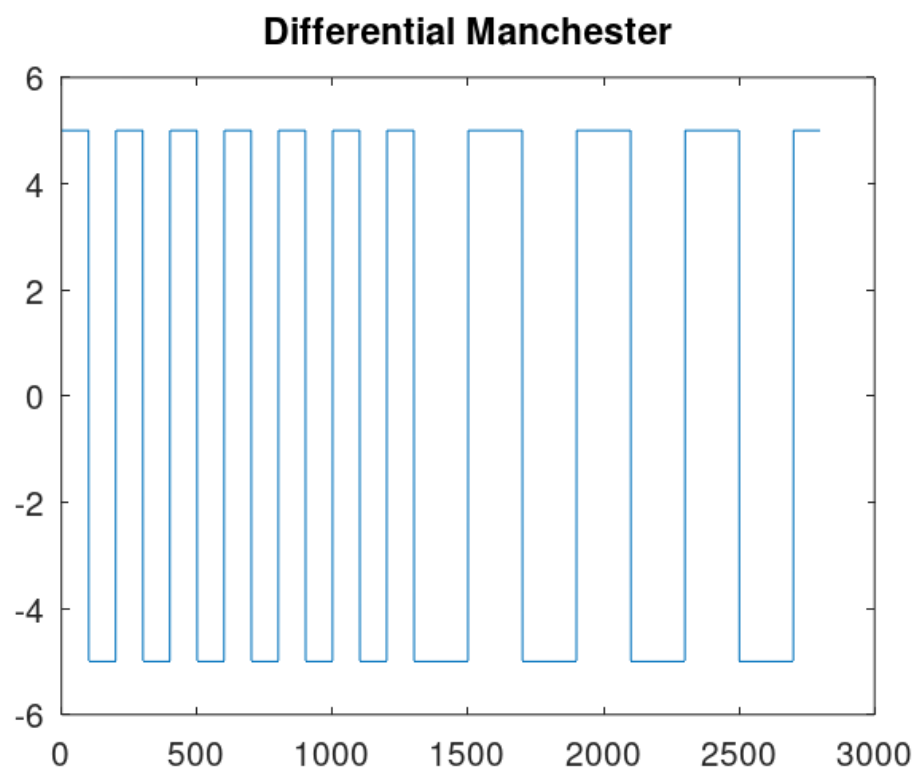


Рис. 2.45: Дифференциальное манчестерское кодирование: есть самосинхронизация

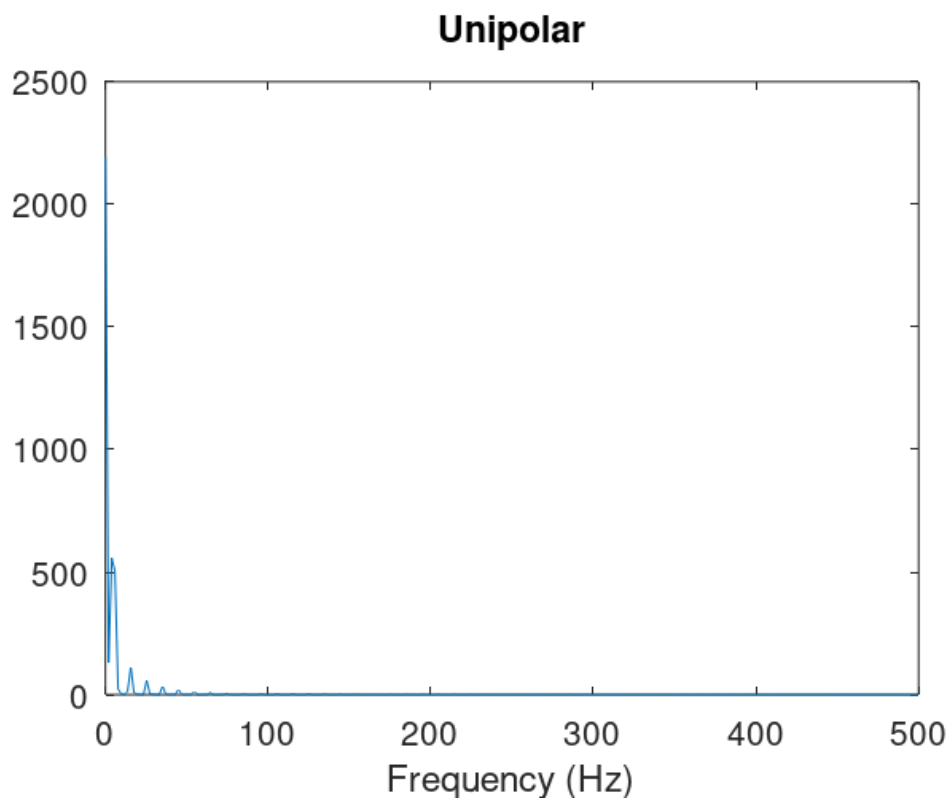


Рис. 2.46: Униполярное кодирование: спектр сигнала

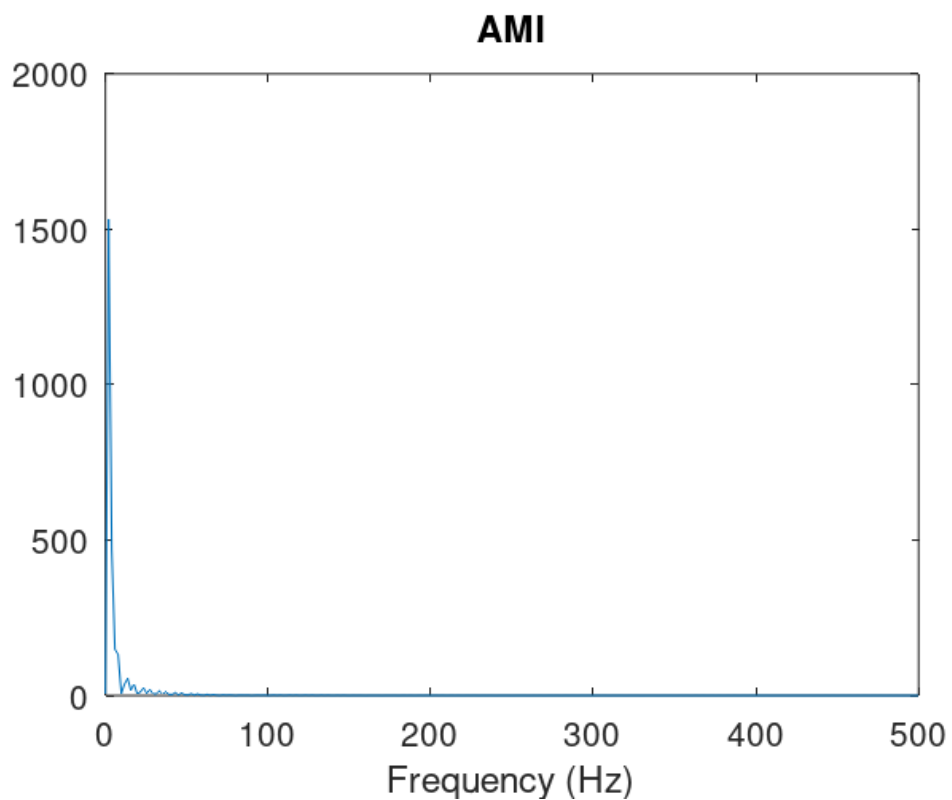


Рис. 2.47: Кодирование AMI: спектр сигнала

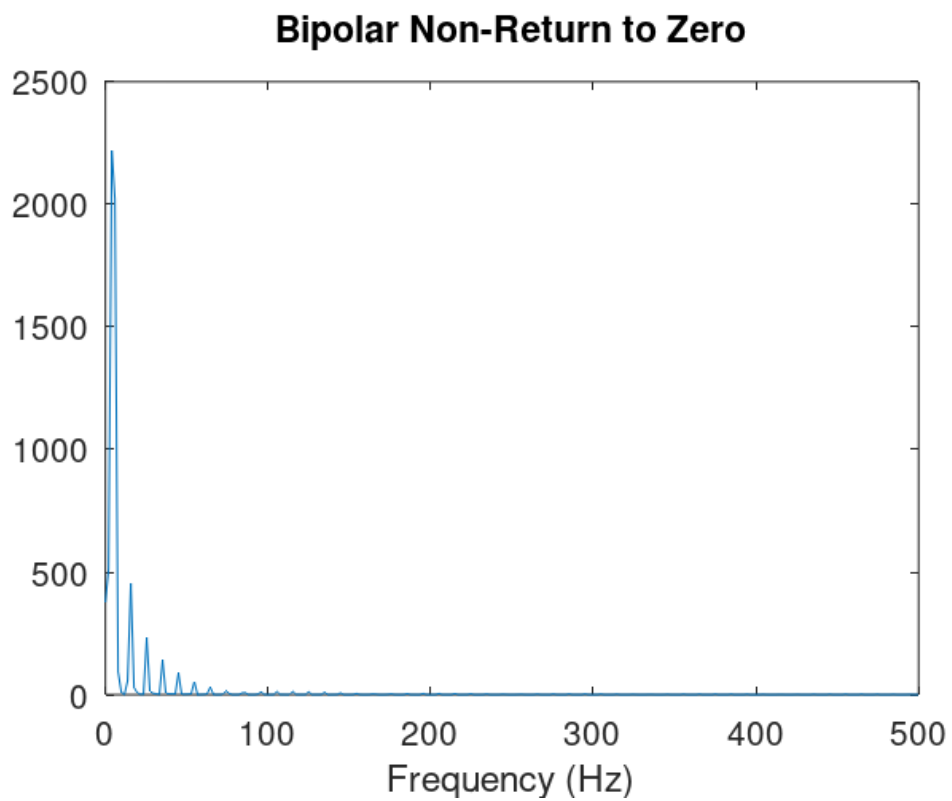


Рис. 2.48: Кодирование NRZ: спектр сигнала

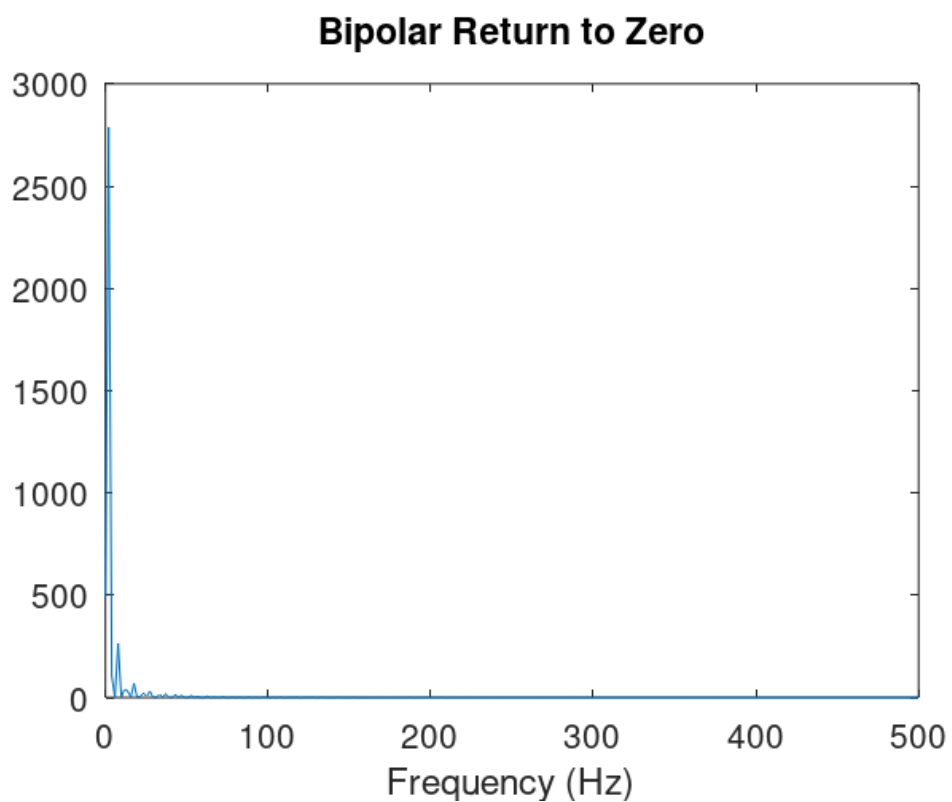


Рис. 2.49: Кодирование RZ: спектр сигнала

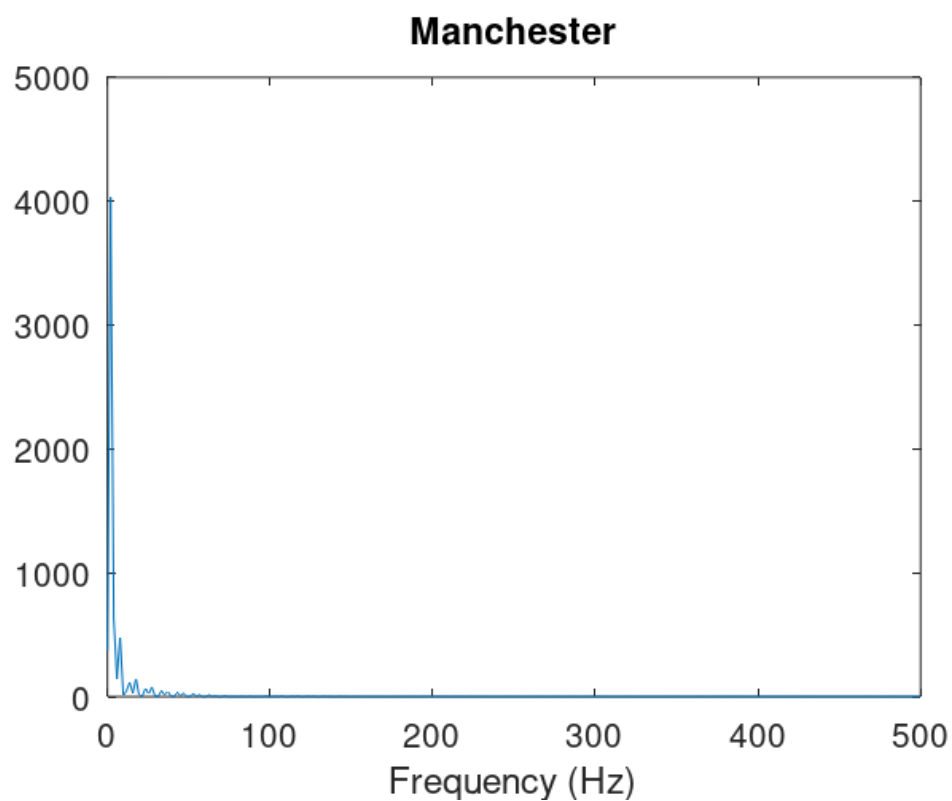


Рис. 2.50: Манчестерское кодирование: спектр сигнала

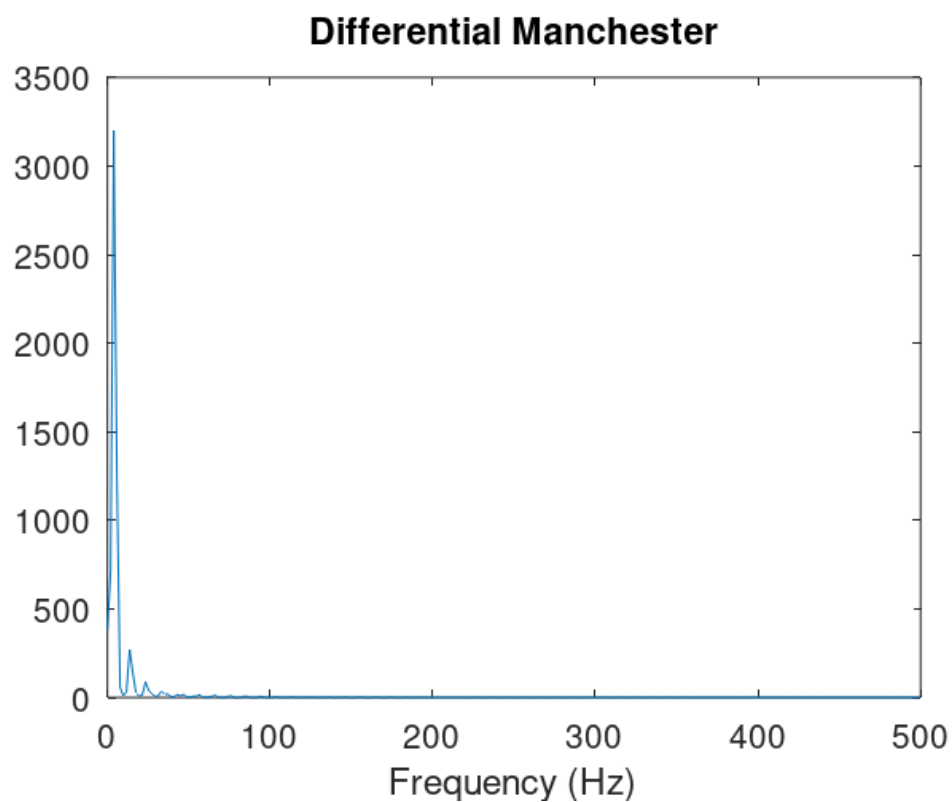


Рис. 2.51: Дифференциальное манчестерское кодирование: спектр сигнала

3 Выводы

В ходе выполнения данной лабораторной работы я изучила методы кодирования и модуляции сигналов с помощью высокоуровневого языка программирования Octave. Определила спектр и параметры сигнала. Продемонстрировала принципы модуляции сигнала на примере аналоговой амплитудной модуляции. Исследовала свойства самосинхронизации сигнала.