

Отчёт по лабораторной работе №1

Дисциплина: Сетевые технологии

Ибрахим Мухсейн Алькамаль

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	11
2.3 Определение спектра и параметров сигнала	14
2.4 Амплитудная модуляция	21
2.5 Кодирование сигнала. Исследование свойства самосинхронизации сигнала	24
3 Выводы	41

Список иллюстраций

2.1	Листинг файла plot_sin.m	8
2.2	График функций y_1 на интервале $-10; 10$	9
2.3	Файлы .eps, .png	9
2.4	Листинг файла plot_sin_cos.m	10
2.5	График функций y_1 и y_2 на интервале $-10; 10$	11
2.6	Листинг файла meandr.m	12
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	25
2.23	Вызовы функций для построения модуляций кодированных сигналов кодовой последовательности data	26
2.24	Вызовы функций для построения модуляций кодированных сигналов кодовой последовательности data_sync	27
2.25	Вызовы функций для построения графиков спектров	28
2.26	Листинг файла unipolar.m	29
2.27	Листинг файла ami.m	29
2.28	Листинг файла bipolarnrz.m	29
2.29	Листинг файла bipolarrrz.m	30
2.30	Листинг файла manchester.m	30
2.31	Листинг файла diffmanc.m	30
2.32	Листинг файла calcspectre.m	31
2.33	Униполярное кодирование	32
2.34	Кодирование AMI	32

2.35 Кодирование NRZ	33
2.36 Кодирование RZ	33
2.37 Манчестерское кодирование	34
2.38 Дифференциальное манчестерское кодирование	34
2.39 Униполярное кодирование: нет самосинхронизации	35
2.40 Кодирование AMI: самосинхронизация при наличии сигнала	35
2.41 Кодирование NRZ: нет самосинхронизации	36
2.42 Кодирование RZ: есть самосинхронизация	36
2.43 Манчестерское кодирование: есть самосинхронизация	37
2.44 Дифференциальное манчестерское кодирование: есть самосинхронизация	37
2.45 Униполярное кодирование: спектр сигнала	38
2.46 Кодирование AMI: спектр сигнала	38
2.47 Кодирование NRZ: спектр сигнала	39
2.48 Кодирование RZ: спектр сигнала	39
2.49 Манчестерское кодирование: спектр сигнала	40
2.50 Дифференциальное манчестерское кодирование: спектр сигнала . . .	40

Список таблиц

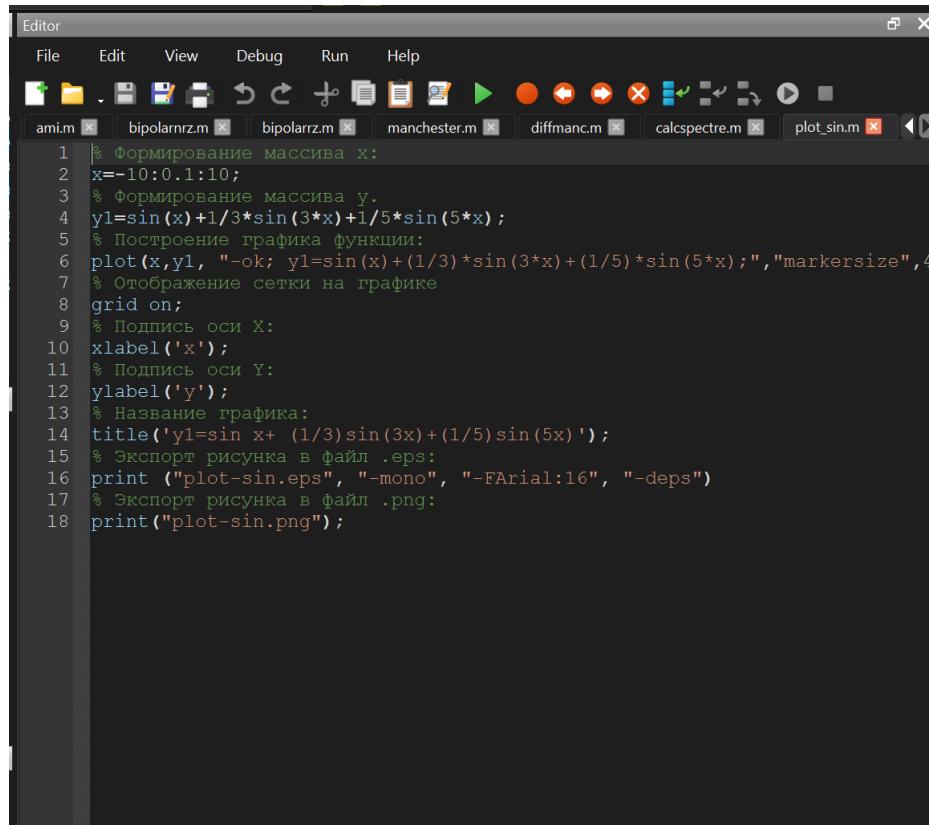
1 Цель работы

Целью данной работы является изучение методов кодирования и модуляции сигналов с помощью высокогоуровневого 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 $y = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$ на интервале $[-10; 10]$ (рис. [fig:001]).

A screenshot of the MATLAB Editor window. The title bar says "Editor". The menu bar includes "File", "Edit", "View", "Debug", "Run", and "Help". Below the menu is a toolbar with various icons. The main area shows a script named "plot_sin.m" with the following code:

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

Рисунок 2.1: Листинг файла plot_sin.m

Запускаем сценарий на выполнение, открывается окно с графиком (рис. [fig:002]). В рабочем каталоге появляются файла с графиками в форматах .eps, .png (рис. [fig:003]).

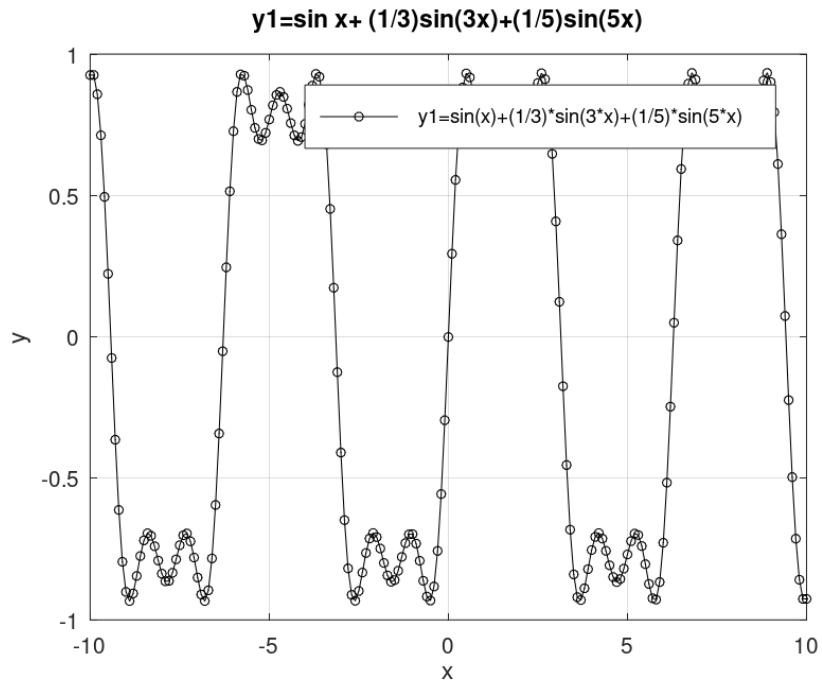


Рисунок 2.2: График функций y_1 на интервале $-10; 10$

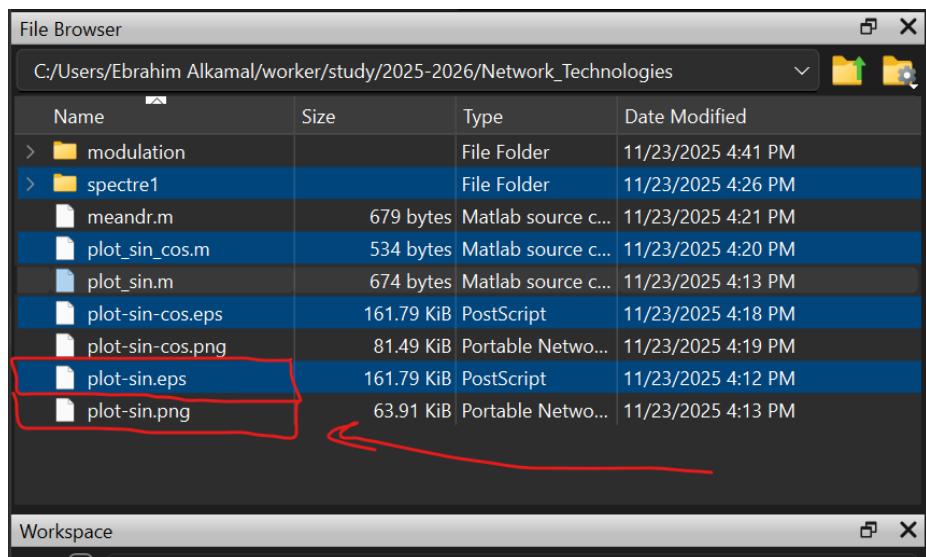
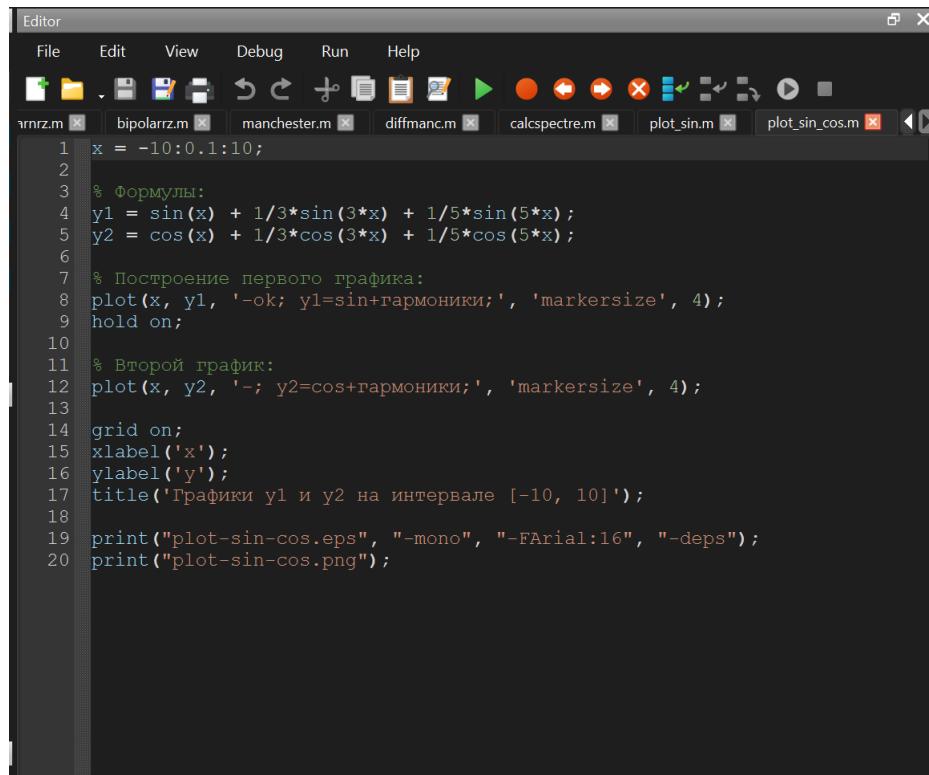


Рисунок 2.3: Файлы .eps, .png

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

листиング (рис. [fig:004]).



The screenshot shows a MATLAB Editor window titled "Editor". The menu bar includes File, Edit, View, Debug, Run, and Help. Below the menu is a toolbar with various icons. The main workspace contains the following MATLAB script:

```
1 x = -10:0.1:10;
2
3 % Формулы:
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
7 % Построение первого графика:
8 plot(x, y1, '-ok; y1=sin+гармоники;', 'markersize', 4);
9 hold on;
10
11 % Второй график:
12 plot(x, y2, '-; y2=cos+гармоники;', 'markersize', 4);
13
14 grid on;
15 xlabel('x');
16 ylabel('y');
17 title('Графики y1 и y2 на интервале [-10, 10]');
18
19 print("plot-sin-cos.eps", "-mono", "-FArial:16", "-deps");
20 print("plot-sin-cos.png");
```

Рисунок 2.4: Листинг файла plot_sin_cos.m

Запускаем, получаем еще один график (рис. [fig:005]).

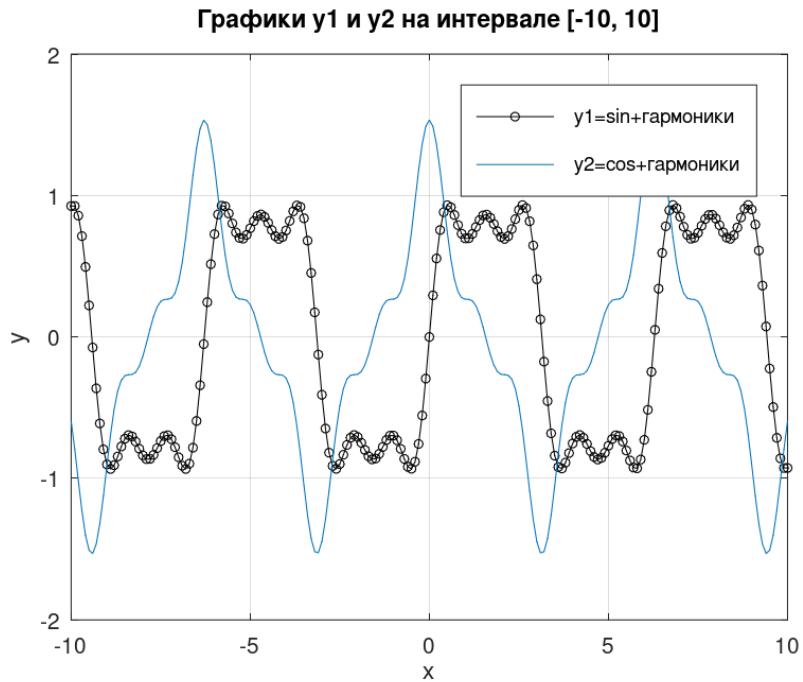


Рисунок 2.5: График функций y_1 и y_2 на интервале $-10; 10$

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. [fig:006]), (рис. [fig:007]).

```

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 % массив гармоник:
16 harmonics=cos(2 * pi * nh' * t/T);
17 % массив элементов ряда:
18 s1=harmonics.*repmat(Am',1,length(t));
19
20 % Суммирование ряда:
21 s2=cumsum(s1);
22 % Построение графиков:
23 for k=1:N
24 subplot(4,2,k)
25 plot(t, s2(k,:))
26 end
27
28 print("plot-meandr-cos.png");

```

Рисунок 2.6: Листинг файла meandr.m

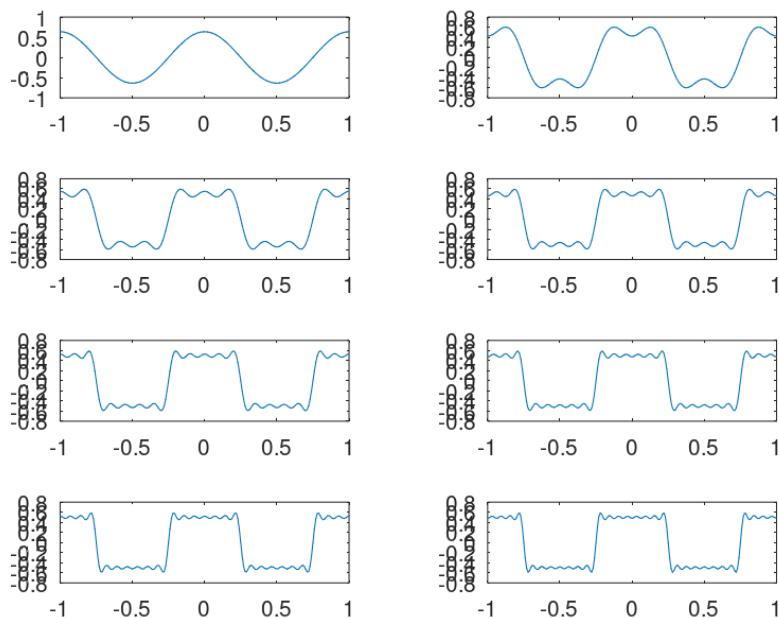
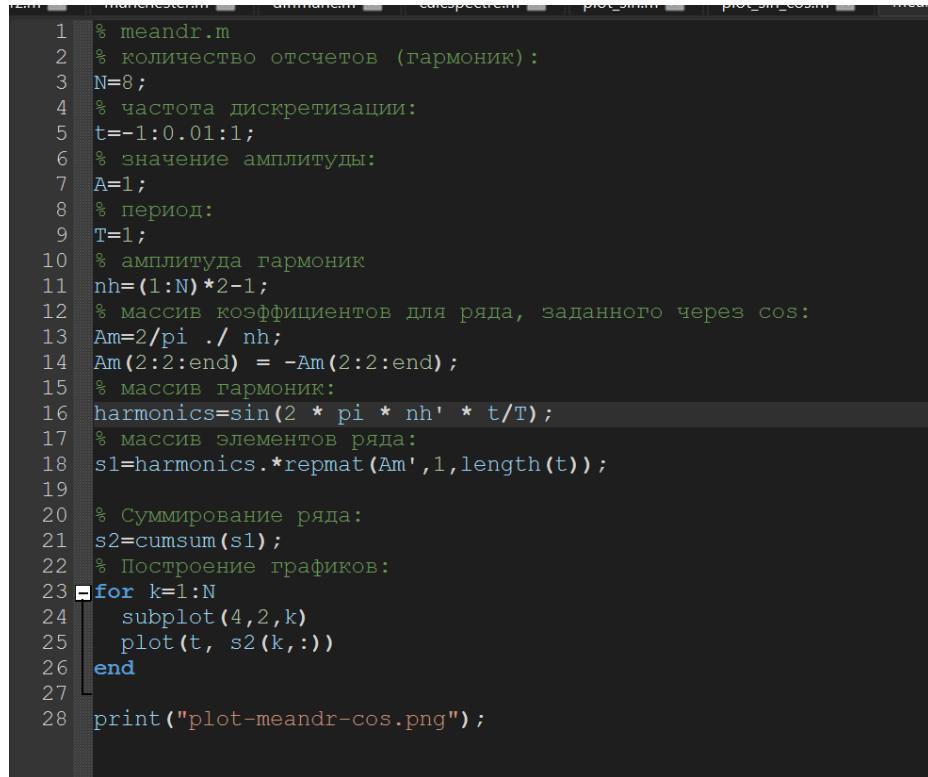


Рисунок 2.7: Меандр через косинусы

Также реализуем меандр через синусы (рис. [fig:008]), (рис. [fig:009]).



```
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 % массив гармоник:
16 harmonics=sin(2 * pi * nh' * t/T);
17 % массив элементов ряда:
18 s1=harmonics.*repmat(Am',1,length(t));
19
20 % Суммирование ряда:
21 s2=cumsum(s1);
22 % Построение графиков:
23 for k=1:N
24 subplot(4,2,k)
25 plot(t, s2(k,:))
26 end
27
28 print("plot-meandr-cos.png");
```

Рисунок 2.8: Листинг файла meandr.m

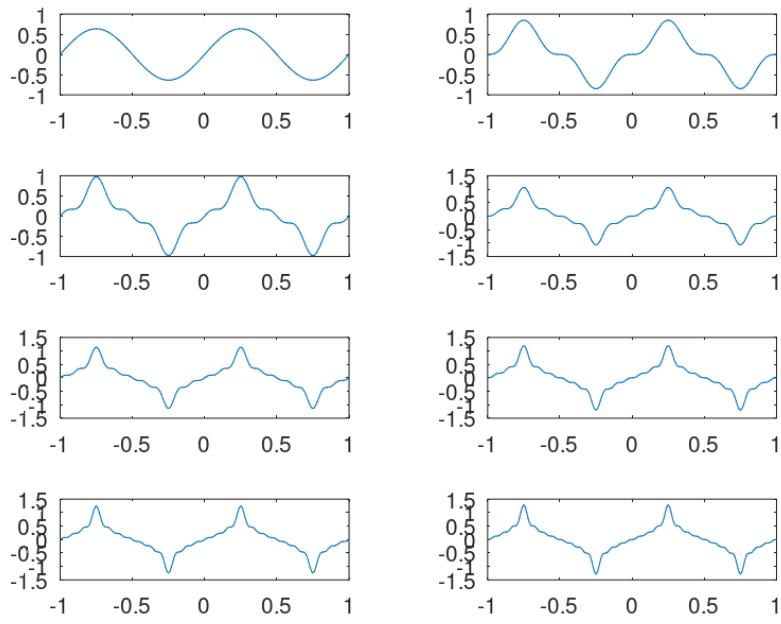


Рисунок 2.9: Меандр через синусы

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. [fig:010]), (рис. [fig:011]).

```
1 % spectrel/spectre.m
2 % Создание каталогов signal и spectre для размещения графиков:
3 mkdir 'signal';
4 mkdir 'spectre';
5 % Длина сигнала (с):
6 tmax = 0.5;
7 % Частота дискретизации (Гц) (количество отсчётов):
8 fd = 512;
9 % Частота первого сигнала (Гц):
10 f1 = 10;
11 % Частота второго сигнала (Гц):
12 f2 = 40;
13 % Амплитуда первого сигнала:
14 a1 = 1;
15 % Амплитуда второго сигнала:
16 a2 = 0.7;
17 % Массив отсчётов времени:
18 t = 0:1./fd:tmax;
19 % Спектр сигнала:
20 fd2 = fd/2;
21 % Два сигнала разной частоты:
22 signal1 = a1*sin(2*pi*t*f1);
23 signal2 = a2*sin(2*pi*t*f2);
24
25 % График 1-го сигнала:
26 plot(signal1,'b');
27 % График 2-го сигнала:
28 hold on
29 plot(signal2,'r');
30 hold off
31 title('Signal');
32 % Экспорт графика в файл в каталоге signal:
33 print 'signal/spectre.png';
34
```

Рисунок 2.10: Листинг файла spectre.m

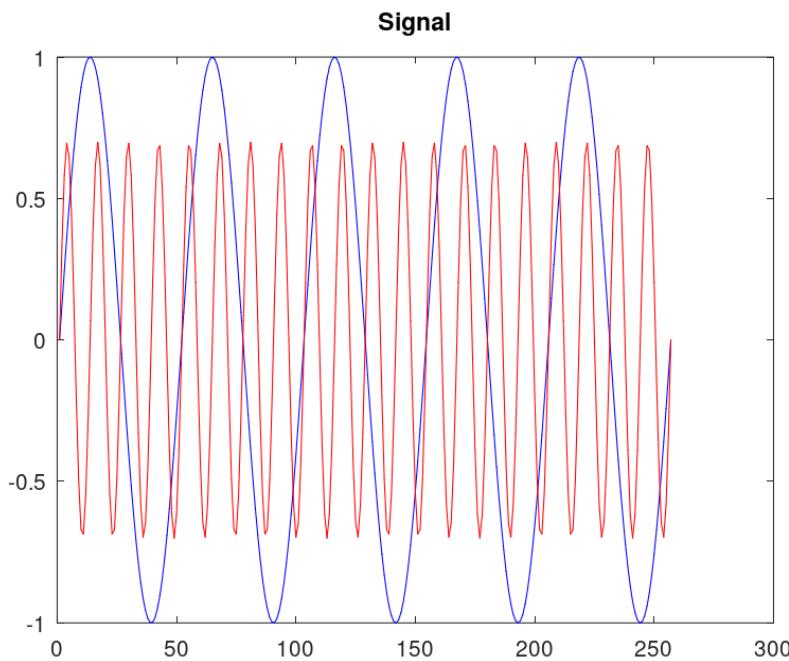


Рисунок 2.11: Графики сигналов разной частоты

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. [fig:012]): отбросим дублирующие отрицательные частоты, а также примем в расчёт то, что на каждом шаге вычисления быстрого преобразования Фурье происходит суммирование амплитуд сигналов.

```

34
35 % Посчитаем спектр
36 % Амплитуды преобразования Фурье сигнала 1:
37 spectre1 = abs(fft(signal1,fd));
38 % Амплитуды преобразования Фурье сигнала 2:
39 spectre2 = abs(fft(signal2,fd));
40 % Построение графиков спектров сигналов:
41 plot(spectre1,'b');
42 hold on
43 plot(spectre2,'r');
44 hold off
45 title('Spectre');
46 print 'spectre/spectre.png';
47
48 % Исправление графика спектра
49 % Сетка частот:
50 f = 1000*(0:fd2)./(2*fd);
51 % Нормировка спектров по амплитуде:
52 spectre1 = 2*spectre1/fd2;
53 spectre2 = 2*spectre2/fd2;
54 % Построение графиков спектров сигналов:
55 plot(f,spectre1(1:fd2+1),'b');
56 hold on
57
58 plot(f,spectre2(1:fd2+1),'r');
59 hold off
60 xlim([0 100]);
61 title('Fixed spectre');
62 xlabel('Frequency (Hz)');
63 print 'spectre/spectre_fix.png';
64
65

```

Рисунок 2.12: Листинг файла spectre.m

Получим следующие графики: график спектров синусоидальных сигналов (рис. [fig:013]) и исправленный график спектров синусоидальных сигналов (рис. [fig:014]).

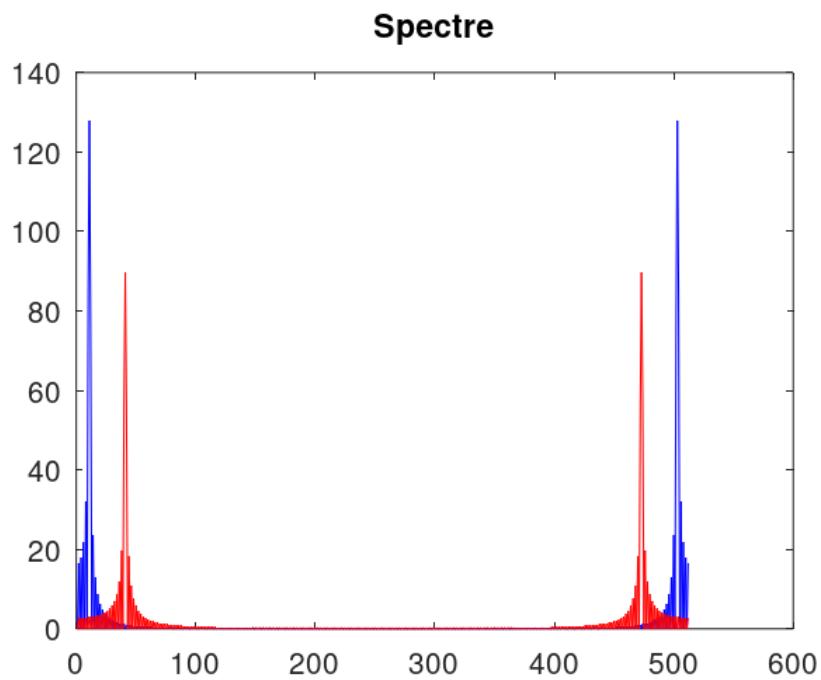


Рисунок 2.13: График спектра синусоидальных сигналов

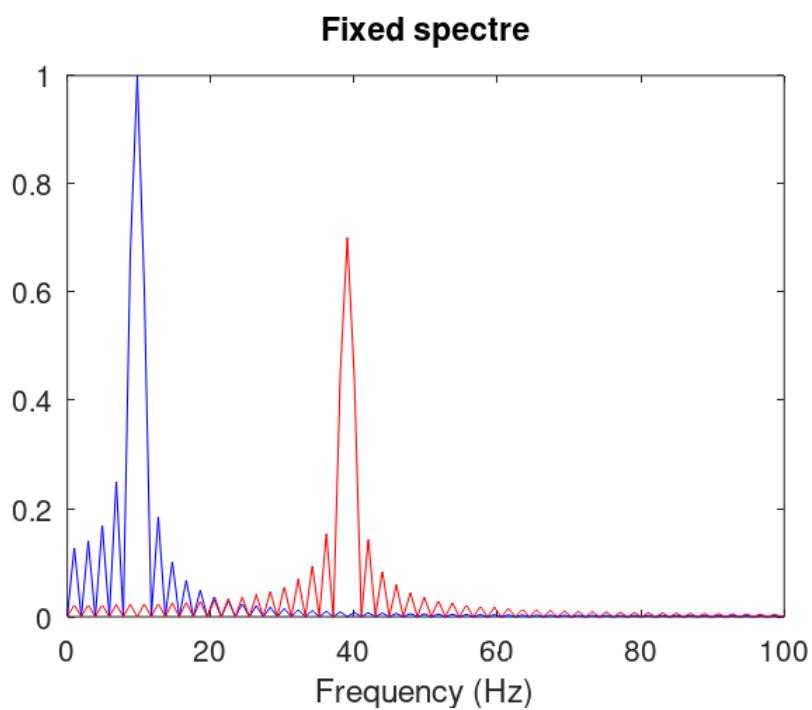
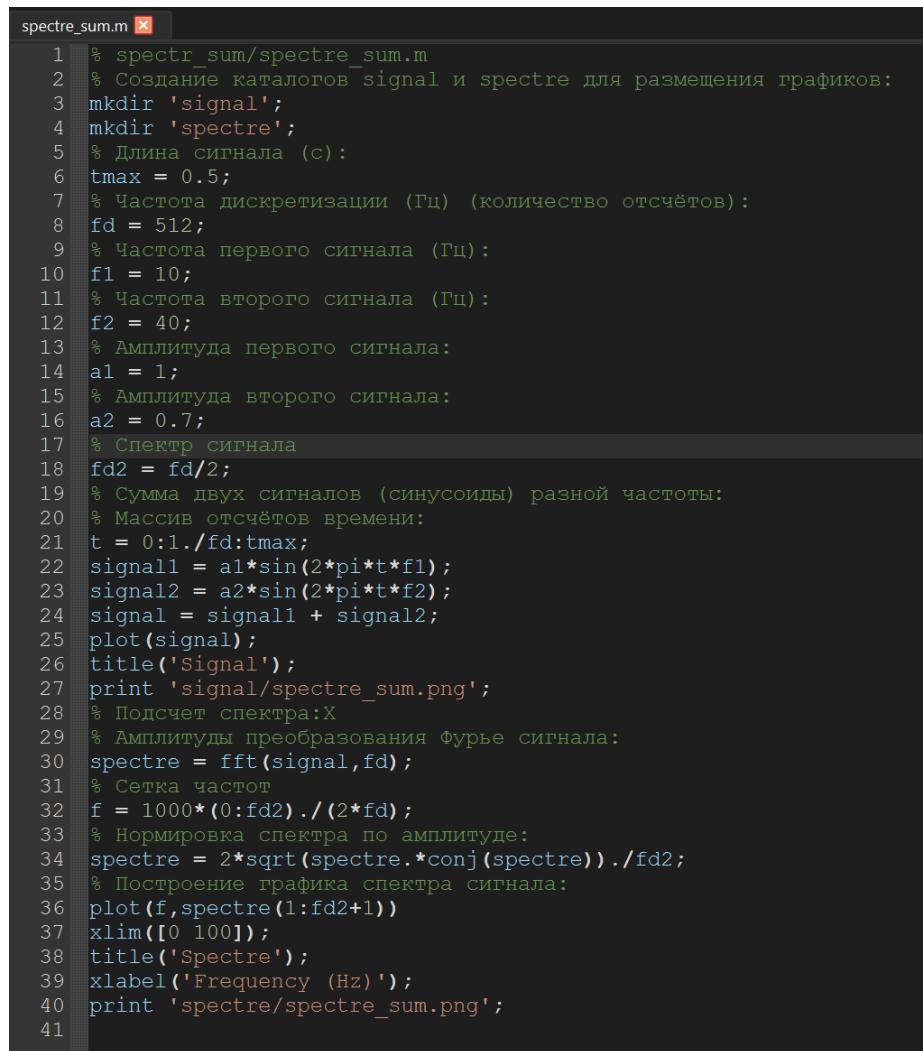


Рисунок 2.14: Исправленный график спектров синусоидальных сигналов

Найдем спектр суммы рассмотренных сигналов, создадим каталог spectr_sum и в нем spectre_sum.m (рис. [fig:015]), (рис. [fig:016]).



```
spectre_sum.m
1 % spectr_sum/spectre_sum.m
2 % Создание каталогов signal и spectre для размещения графиков:
3 mkdir 'signal';
4 mkdir 'spectre';
5 % Длина сигнала (с):
6 tmax = 0.5;
7 % Частота дискретизации (Гц) (количество отсчётов):
8 fd = 512;
9 % Частота первого сигнала (Гц):
10 f1 = 10;
11 % Частота второго сигнала (Гц):
12 f2 = 40;
13 % Амплитуда первого сигнала:
14 a1 = 1;
15 % Амплитуда второго сигнала:
16 a2 = 0.7;
17 % Спектр сигнала
18 fd2 = fd/2;
19 % Сумма двух сигналов (синусоиды) разной частоты:
20 % Массив отсчётов времени:
21 t = 0:1./fd:tmax;
22 signal1 = a1*sin(2*pi*t*f1);
23 signal2 = a2*sin(2*pi*t*f2);
24 signal = signal1 + signal2;
25 plot(signal);
26 title('Signal');
27 print 'signal/spectre_sum.png';
28 % Подсчет спектра:Х
29 % Амплитуды преобразования Фурье сигнала:
30 spectre = fft(signal,fd);
31 % Сетка частот
32 f = 1000*(0:fd2)./(2*fd);
33 % Нормировка спектра по амплитуде:
34 spectre = 2*sqrt(spectre.*conj(spectre))./fd2;
35 % Построение графика спектра сигнала:
36 plot(f,spectre(1:fd2+1))
37 xlim([0 100]);
38 title('Spectre');
39 xlabel('Frequency (Hz)');
40 print 'spectre/spectre_sum.png';
41
```

Рисунок 2.15: Листинг файла spectre_sum.m

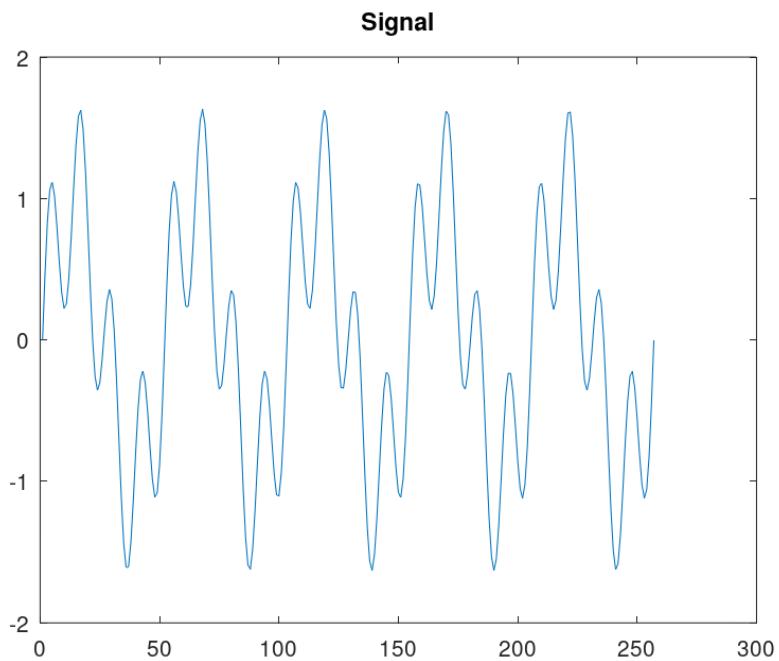


Рисунок 2.16: Суммарный сигнал

В результате должен получится аналогичный предыдущему результат (рис. [fig:017]), т.е. спектр суммы сигналов должен быть равен сумме спектров сигналов, что вытекает из свойств преобразования Фурье.

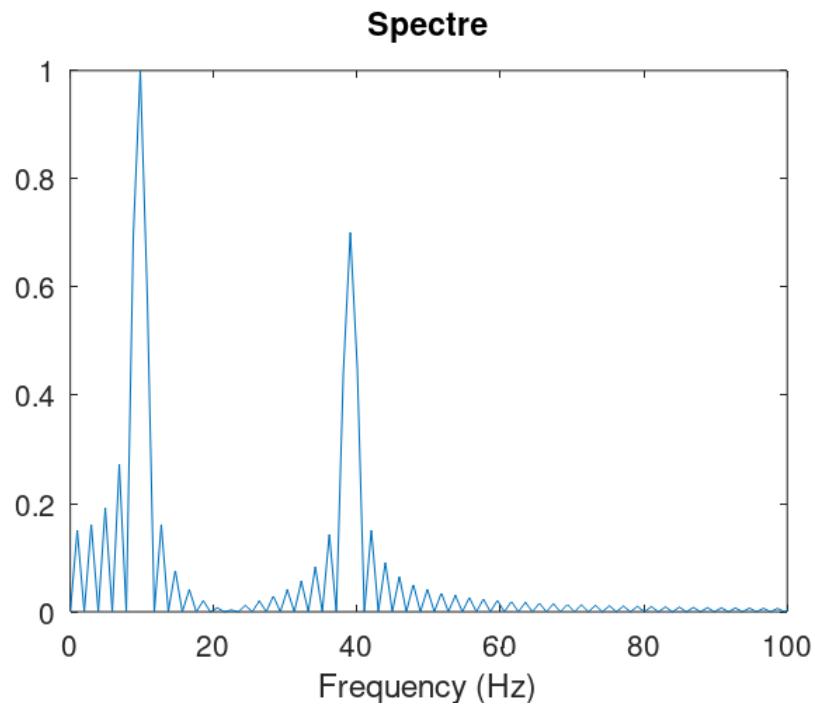


Рисунок 2.17: Спектр суммарного сигнала

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. [fig:018]).

```

am.m x
4 mkdir 'spectre';
5 % Модуляция синусоид с частотами 50 и 5
6 % Длина сигнала (с)
7 tmax = 0.5;
8 % Частота дискретизации (Гц) (количество отсчётов)
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 % разной частоты
18 % Массив отсчётов 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 огибающей:
26 plot(signal1, 'r');
27 plot(-signal1, 'r');
28 hold offZ
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)');
44 print 'spectre/am.png';

```

Рисунок 2.18: Листинг файла am.m

В результате получаем, что спектр произведения представляет собой свертку спектров (рис. [fig:019]), (рис. [fig:020]).

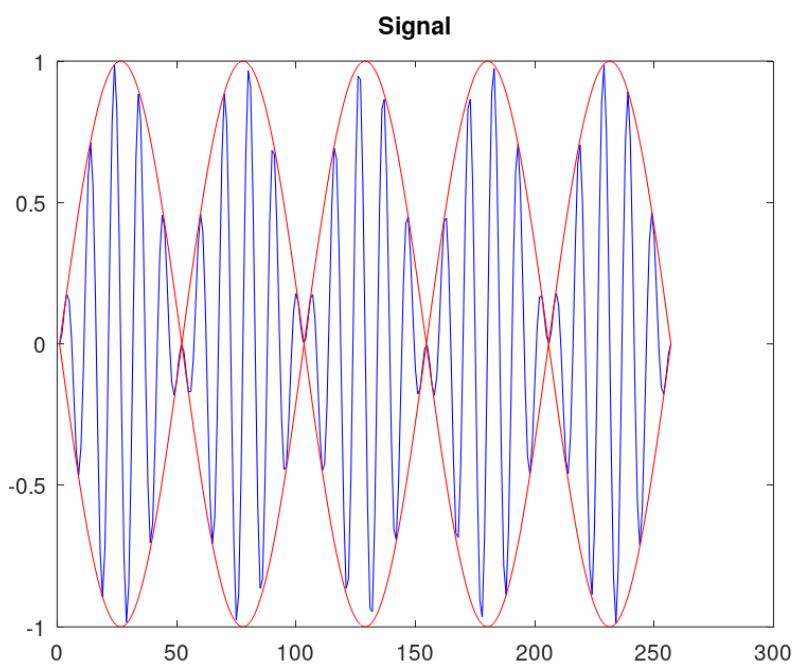


Рисунок 2.19: Сигнал и огибающая при амплитудной модуляции

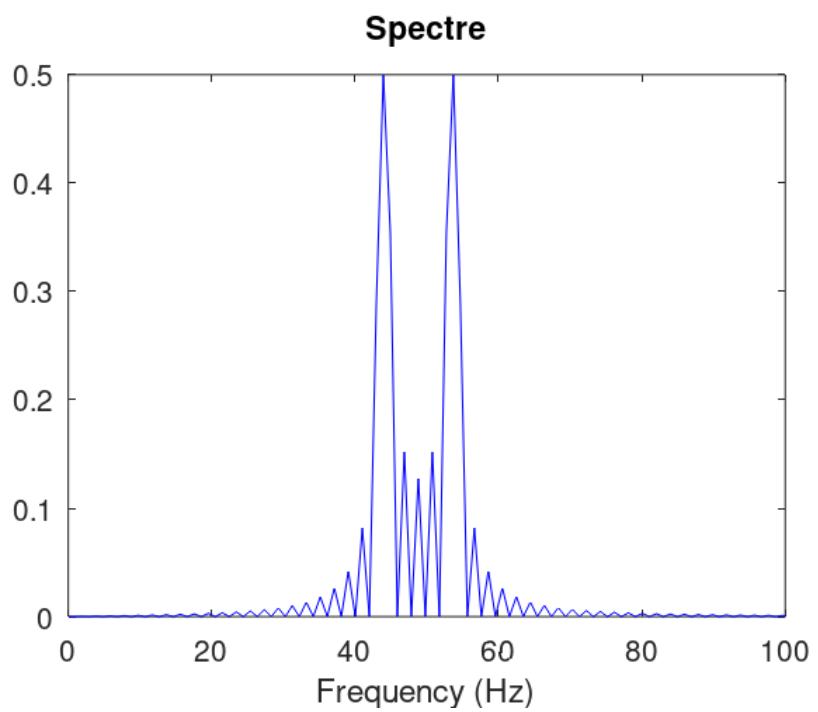


Рисунок 2.20: Спектр сигнала при амплитудной модуляции

2.5 Кодирование сигнала. Исследование свойства самосинхронизации сигнала

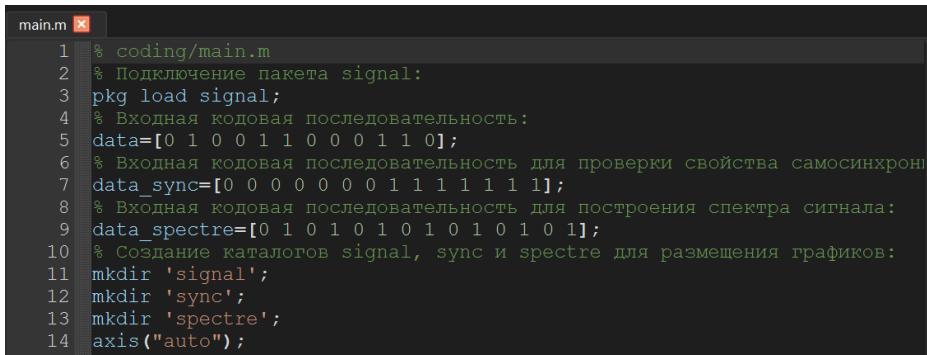
В рабочем каталоге создадим каталог coding и в нём файлы main.m, maptowave.m, unipolar.m, ami.m, bipolarnrz.m, bipolarrrz.m, manchester.m, diffmanc.m, calcspctre.m.

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. [fig:021]).

Package Name	Version	Installation directory
audio	2.0.9	...\Octave-10.3.0\mingw64\share\octave\packages\audio-2.0.9
biosig	3.9.0	...\Octave-10.3.0\mingw64\share\octave\packages\biosig-3.9.0
cfitsio	0.0.7	...\Octave-10.3.0\mingw64\share\octave\packages\cfitsio-0.0.7
coder	1.10.1	...\Octave-10.3.0\mingw64\share\octave\packages\coder-1.10.1
communications	1.2.7	...\mingw64\share\octave\packages\communications-1.2.7
control	4.1.3	...\Octave-10.3.0\mingw64\share\octave\packages\control-4.1.3
data-smoothing	1.3.0	...\mingw64\share\octave\packages\data-smoothing-1.3.0
database	2.4.4	...\Octave-10.3.0\mingw64\share\octave\packages\database-2.4.4
dataframe	1.2.0	...\mingw64\share\octave\packages\dataframe-1.2.0
dicom	0.6.1	...\Octave-10.3.0\mingw64\share\octave\packages\dicom-0.6.1
financial	0.5.4	...\mingw64\share\octave\packages\financial-0.5.4
fl-core	1.0.2	...\Octave-10.3.0\mingw64\share\octave\packages\fl-core-1.0.2
fuzzy-logic-toolkit	0.6.2	...\mingw64\share\octave\packages\fuzzy-logic-toolkit-0.6.2
ga	0.10.4	...\Octave-10.3.0\mingw64\share\octave\packages\ga-0.10.4
general	2.1.3	...\Octave-10.3.0\mingw64\share\octave\packages\general-2.1.3
generate_html	0.3.3	...\mingw64\share\octave\packages\generate_html-0.3.3
geometry	4.1.0	...\Octave-10.3.0\mingw64\share\octave\packages\geometry-4.1.0
gsl	2.1.1	...\Octave-10.3.0\mingw64\share\octave\packages\gsl-2.1.1
image	2.18.1	...\Octave-10.3.0\mingw64\share\octave\packages\image-2.18.1
image-acquisition	0.3.3	...\mingw64\share\octave\packages\image-acquisition-0.3.3
instrument-control	0.9.5	...\mingw64\share\octave\packages\instrument-control-0.9.5
interval	3.2.1	...\Octave-10.3.0\mingw64\share\octave\packages\interval-3.2.1
io	2.7.0	...\Octave-10.3.0\mingw64\share\octave\packages\io-2.7.0
linear-algebra	2.2.3	...\mingw64\share\octave\packages\linear-algebra-2.2.3
lssa	0.1.4	...\Octave-10.3.0\mingw64\share\octave\packages\lssa-0.1.4
ltfat	2.6.0	...\Octave-10.3.0\mingw64\share\octave\packages\ltfat-2.6.0
mapping	1.4.3	...\Octave-10.3.0\mingw64\share\octave\packages\mapping-1.4.3
matgeom	1.2.4	...\Octave-10.3.0\mingw64\share\octave\packages\matgeom-1.2.4
miscellaneous	1.3.1	...\mingw64\share\octave\packages\miscellaneous-1.3.1
mqtt	0.0.5	...\Octave-10.3.0\mingw64\share\octave\packages\mqtt-0.0.5
nan	3.7.0	...\Octave-10.3.0\mingw64\share\octave\packages\nan-3.7.0
netcdf	1.0.18	...\Octave-10.3.0\mingw64\share\octave\packages\ncdf-1.0.18
nurbs	1.4.4	...\Octave-10.3.0\mingw64\share\octave\packages\nurbs-1.4.4
ocs	0.1.5	...\Octave-10.3.0\mingw64\share\octave\packages\ocs-0.1.5
octproj	3.1.0	...\Octave-10.3.0\mingw64\share\octave\packages\octproj-3.1.0
optim	1.6.2	...\Octave-10.3.0\mingw64\share\octave\packages\optim-1.6.2
optiminterp	0.3.7	...\mingw64\share\octave\packages\optiminterp-0.3.7
parallel	4.0.2	...\Octave-10.3.0\mingw64\share\octave\packages\parallel-4.0.2
quaternion	2.4.0	...\mingw64\share\octave\packages\quaternion-2.4.0
queueing	1.2.8	...\Octave-10.3.0\mingw64\share\octave\packages\queueing-1.2.8
signal	1.4.6	...\Octave-10.3.0\mingw64\share\octave\packages\signal-1.4.6

Рисунок 2.21: Проверка правильности установки пакета signal

В файле main.m подключаем пакет signal и задаем входные кодовые последовательности (рис. [fig:022]).



```
main.m
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 самосинхронии:
7 data_sync=[0 0 0 0 0 0 1 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");
```

Рисунок 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. [fig:023]).

```

11 axis(data);
15 % Унипольярное кодирование
16 wave=unipolar(data);
17 plot(wave);
18 ylim([-1 6]);
19 title('Unipolar');
20 print 'signal/unipolar.png';
21 % Кодирование ami
22 wave=ami(data);
23 plot(wave)
24 title('AMI');
25 print 'signal/ami.png';
26 % Кодирование NRZ
27 wave=bipolarnrz(data);
28 plot(wave);
29 title('Bipolar Non-Return to Zero');
30 print 'signal/bipolarnrz.png';
31 % Кодирование RZ
32 wave=bipolarrz(data);
33 plot(wave)
34 title('Bipolar Return to Zero');
35 print 'signal/bipolarrz.png';
36 % Манчестерское кодирование
37 wave=manchester(data);
38 plot(wave)
39 title('Manchester');
40 print 'signal/manchester.png';
41
42 % Дифференциальное манчестерское кодирование
43 wave=diffmanc(data);
44 plot(wave)
45 title('Differential Manchester');
46 print 'signal/diffmanc.png';

```

Рисунок 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. [fig:024]).

```
main.m x
47 % Униполярное кодирование
48 wave=unipolar(data_sync);
49 plot(wave);
50 ylim([-1 6]);
51 title('Unipolar');
52 print 'sync/unipolar.png';
53 % Кодирование AMI
54 wave=ami(data_sync);
55 plot(wave)
56 title('AMI');
57 print 'sync/ami.png';
58 % Кодирование NRZ
59 wave=bipolarnrz(data_sync);
60 plot(wave);
61 title('Bipolar Non-Return to Zero');
62 print 'sync/bipolarnrz.png';
63 % Кодирование RZ
64 wave=bipolarrz(data_sync);
65 plot(wave)
66 title('Bipolar Return to Zero');
67 print 'sync/bipolarrz.png';
68 % Манчестерское кодирование
69 wave=manchester(data_sync);
70 plot(wave)
71 title('Manchester');
72 print 'sync/manchester.png';
73 % Дифференциальное манчестерское кодирование
74 wave=diffmanc(data_sync);
75 plot(wave)
76 title('Differential Manchester');
77 print 'sync/diffmanc.png';
```

Рисунок 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. [fig:025]).

```

78 % Унипольярное кодирование:
79 wave=unipolar(data_spectre);
80 spectre=calcspectre(wave);
81 title('Unipolar');
82 print 'spectre/unipolar.png';
83 % Кодирование AMI:
84 wave=ami(data_spectre);
85 spectre=calcspectre(wave);
86 title('AMI');
87 print 'spectre/ami.png';
88 % Кодирование NRZ:
89 wave=bipolarnrz(data_spectre);
90 spectre=calcspectre(wave);
91 title('Bipolar Non-Return to Zero');
92 print 'spectre/bipolarnrz.png';
93 % Кодирование RZ:
94 wave=bipolarrz(data_spectre);
95 spectre=calcspectre(wave);
96 title('Bipolar Return to Zero');
97 print 'spectre/bipolarrz.png';
98 % Манчестерское кодирование:
99 wave=manchester(data_spectre);
100 spectre=calcspectre(wave);
101 title('Manchester');
102 print 'spectre/manchester.png';
103 % Дифференциальное манчестерское кодирование:
104 wave=diffmanc(data_spectre);
105 spectre=calcspectre(wave);
106 title('Differential Manchester');
107 print 'spectre/diffmanc.png';
108

```

Рисунок 2.25: Вызовы функций для построения графиков спектров

В файле maptowave.m пропишем функцию, которая по входному битовому потоку строит график сигнала (рис. [fig:026]).

В файлах unipolar.m (рис. [fig:027]), ami.m (рис. [fig:028]), bipolarnrz.m (рис. [fig:029]), bipolarrz.m (рис. [fig:030]), manchester.m (рис. [fig:031]), diffmanc.m (рис. [fig:032]) пропишем соответствующие функции преобразования кодовой последовательности data с вызовом функции maptowave для построения соответствующего графика.

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

Рисунок 2.26: Листинг файла unipolar.m

```
unipolar.m ami.m
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);
8 end
```

Рисунок 2.27: Листинг файла ami.m

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

Рисунок 2.28: Листинг файла bipolarnrz.m

```
bipolarrz.m
1 % coding/bipolarrz.m
2 % Кодирование RZ:
3 function wave=bipolarrz(data)
4     data(data==0)=-1;
5     data=upsample(data,2);
6     wave=maptowave(data);
7 end
```

Рисунок 2.29: Листинг файла bipolarrz.m

```
bipolarrz.m manchester.m
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);
8 end
```

Рисунок 2.30: Листинг файла manchester.m

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

Рисунок 2.31: Листинг файла diffmanc.m

В файле calcspectre.m пропишем функцию построения спектра сигнала (рис. [fig:033]).

The screenshot shows a MATLAB code editor window with two tabs: 'diffmanc.m' and 'calcspectre.m'. The 'calcspectre.m' tab is active, displaying the following MATLAB script:

```
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;
10    f = 1000*(0:Fd2)/Fd;
11    plot(f,spectre(1:Fd3));
12    xlabel('Frequency (Hz)');
13 end
```

Рисунок 2.32: Листинг файла calcspectre.m

Запустим главный скрипт main.m. В каталоге signal должны быть получены файлы с графиками кодированного сигнала (рис. [fig:034]-[fig:039]), в каталоге sync – файлы с графиками, иллюстрирующими свойства самосинхронизации (рис. [fig:040]-[fig:045]), в каталоге spectre – файлы с графиками спектров сигналов (рис. [fig:046]-[fig:051]).

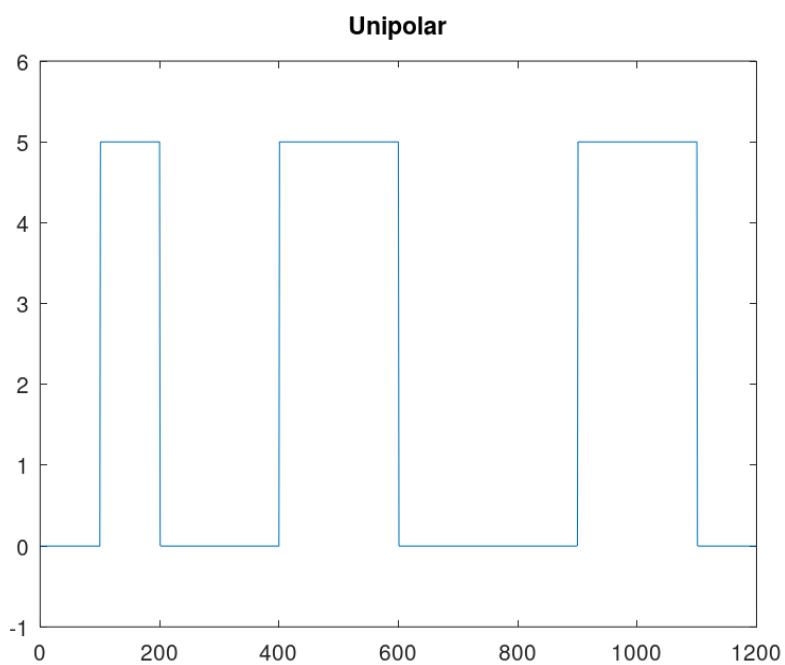


Рисунок 2.33: Униполярное кодирование

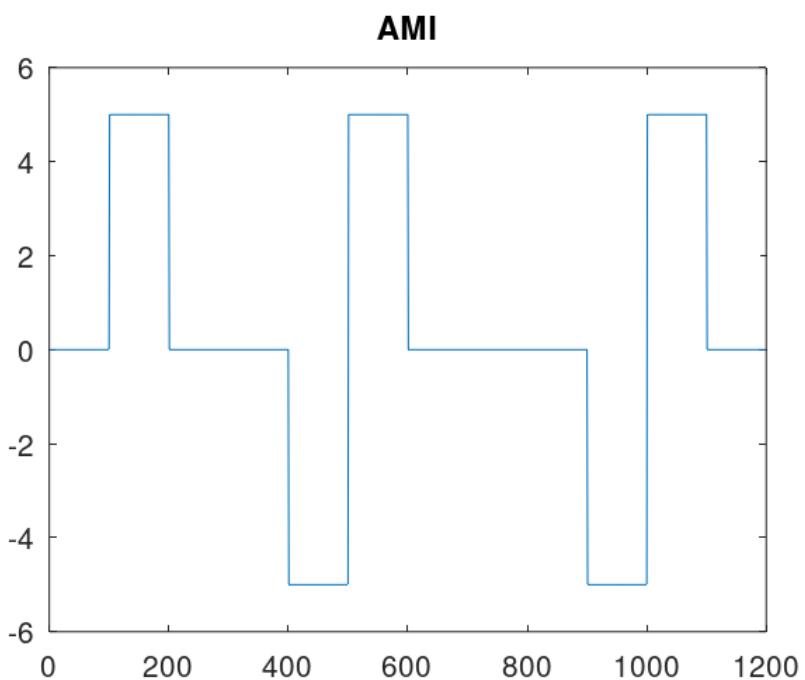


Рисунок 2.34: Кодирование AMI

Bipolar Non-Return to Zero

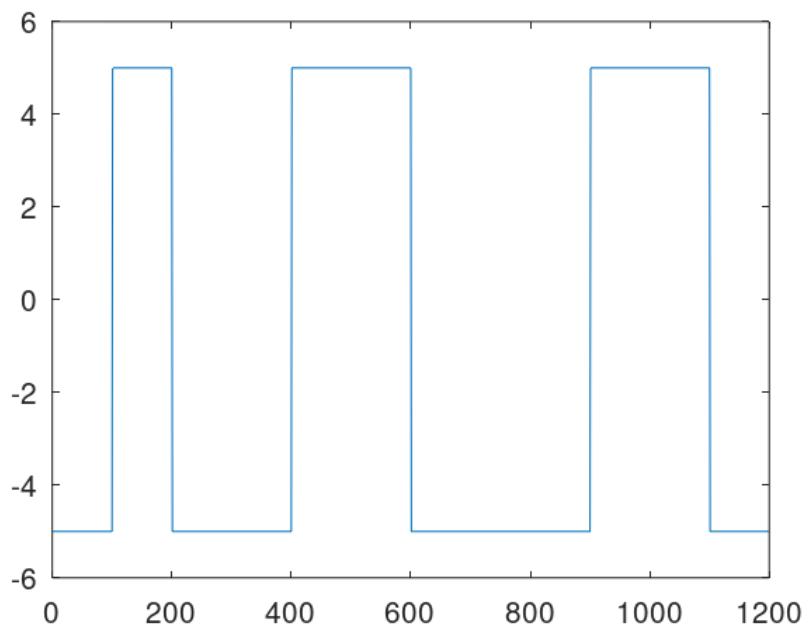


Рисунок 2.35: Кодирование NRZ

Bipolar Return to Zero

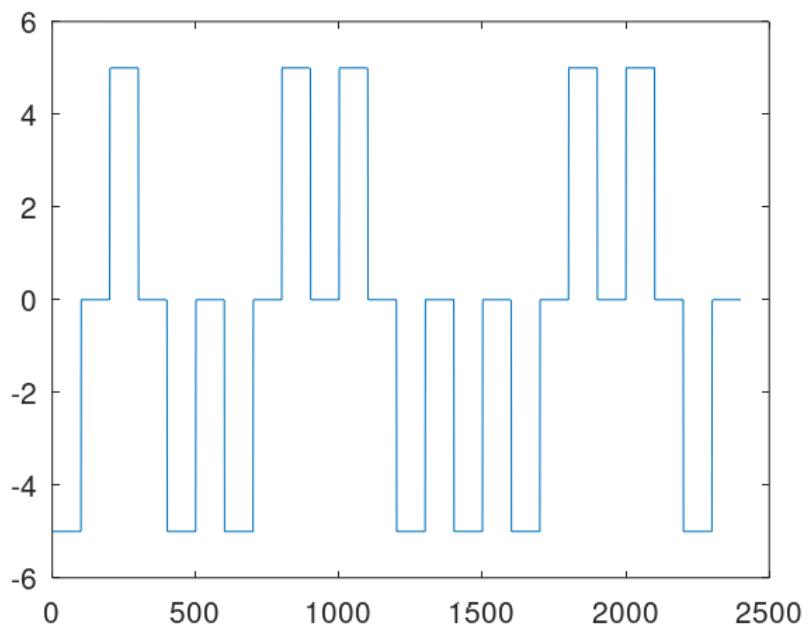


Рисунок 2.36: Кодирование RZ

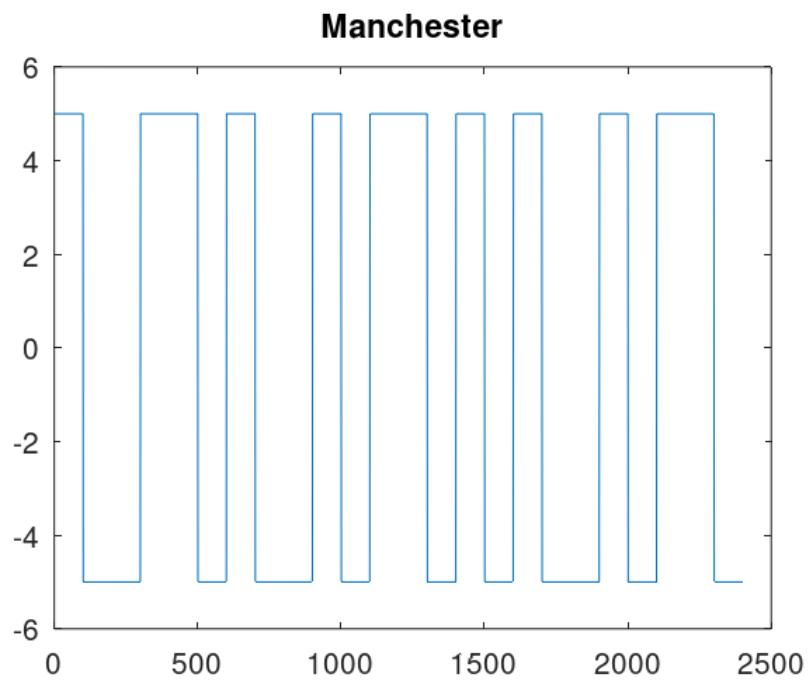


Рисунок 2.37: Манчестерское кодирование

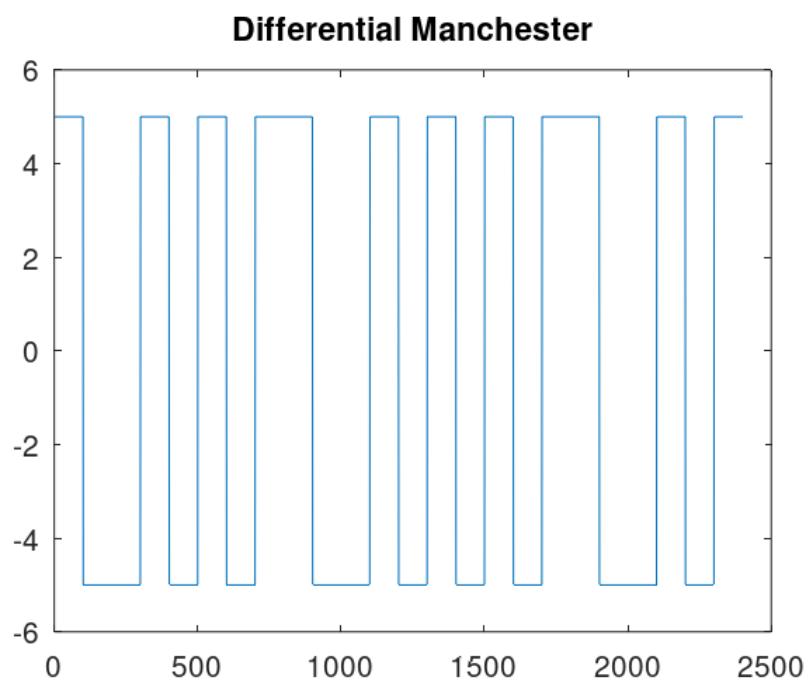


Рисунок 2.38: Дифференциальное манчестерское кодирование

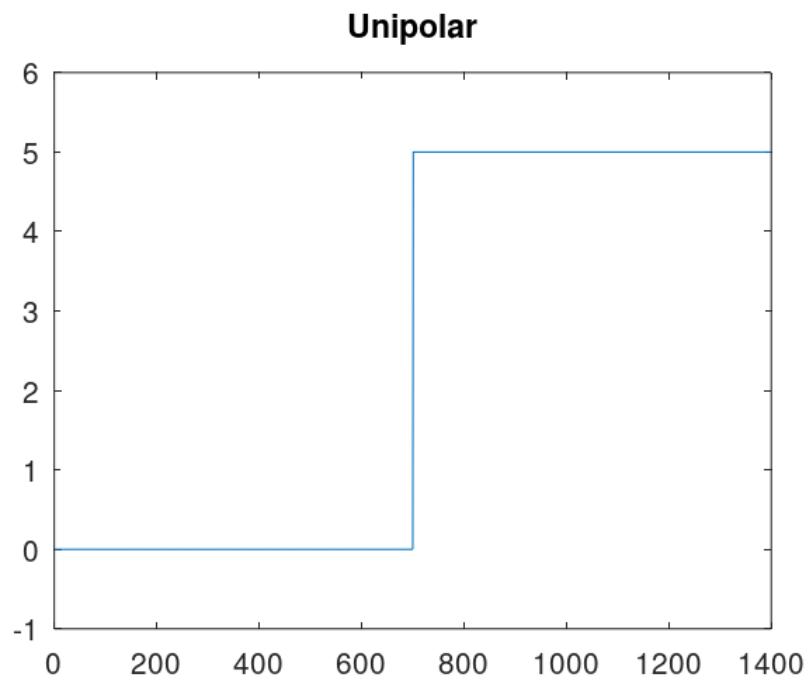


Рисунок 2.39: Униполярное кодирование: нет самосинхронизации

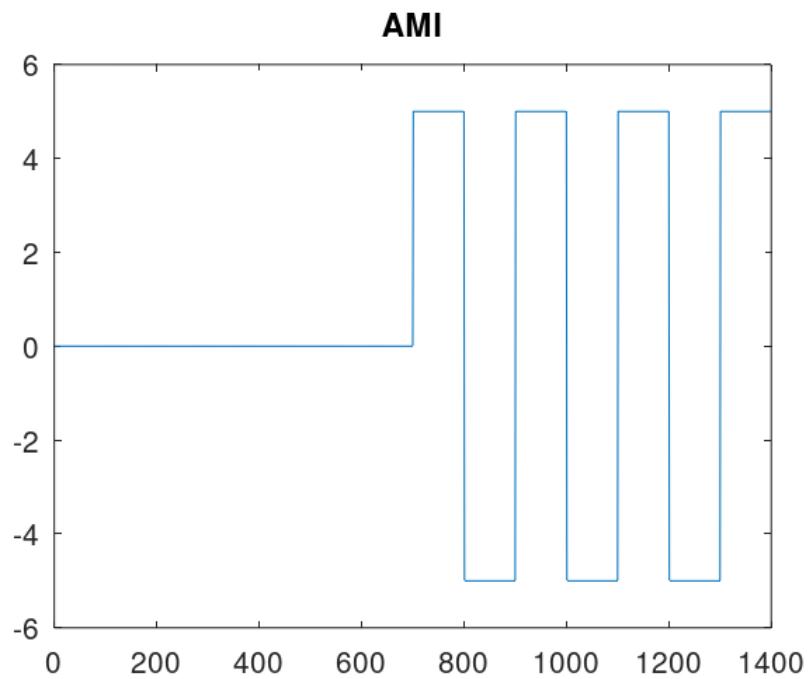


Рисунок 2.40: Кодирование AMI: самосинхронизация при наличии сигнала

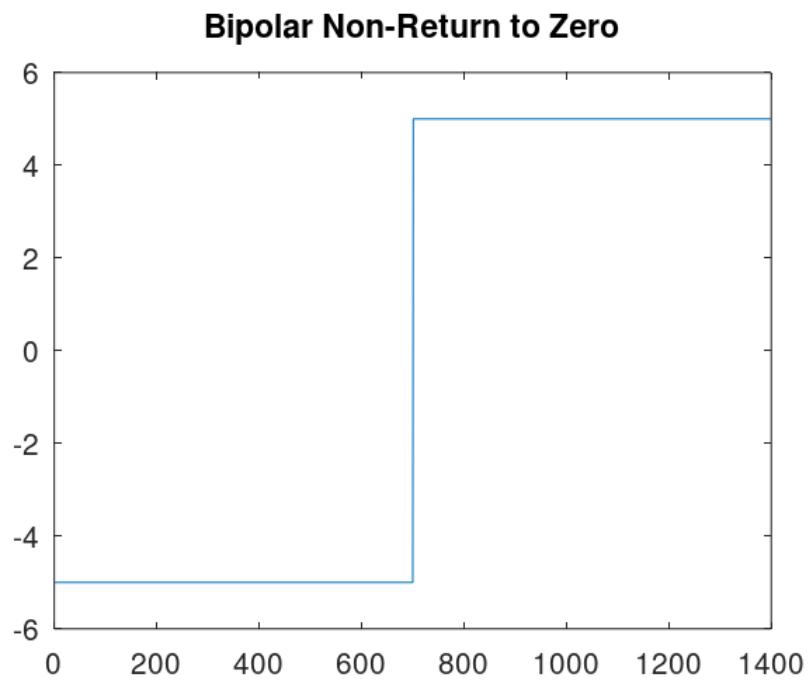


Рисунок 2.41: Кодирование NRZ: нет самосинхронизации

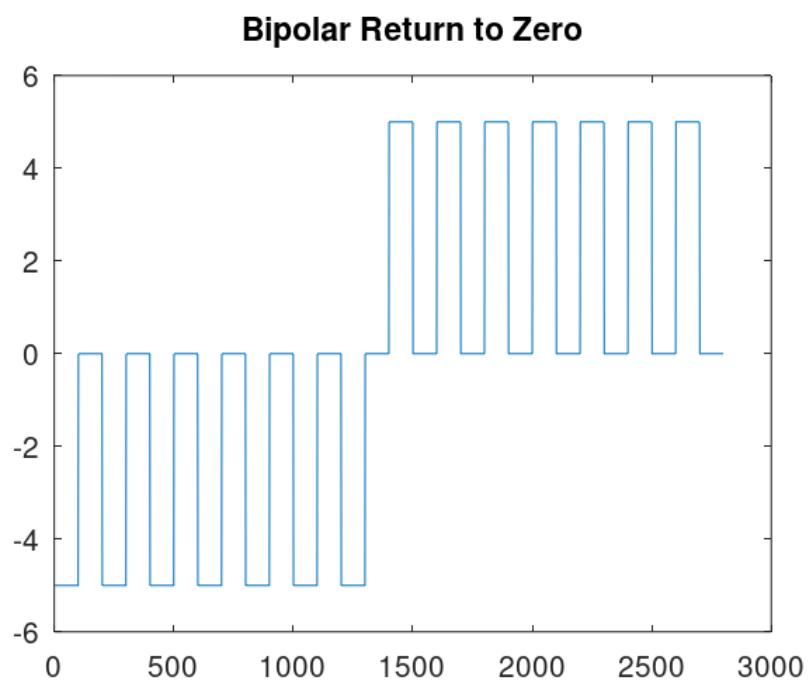


Рисунок 2.42: Кодирование RZ: есть самосинхронизация

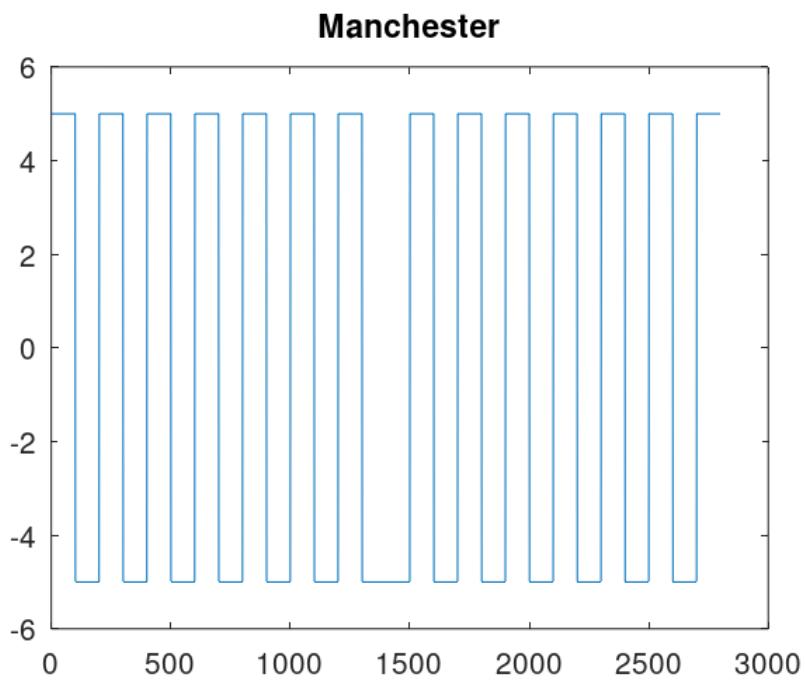


Рисунок 2.43: Манчестерское кодирование: есть самосинхронизация

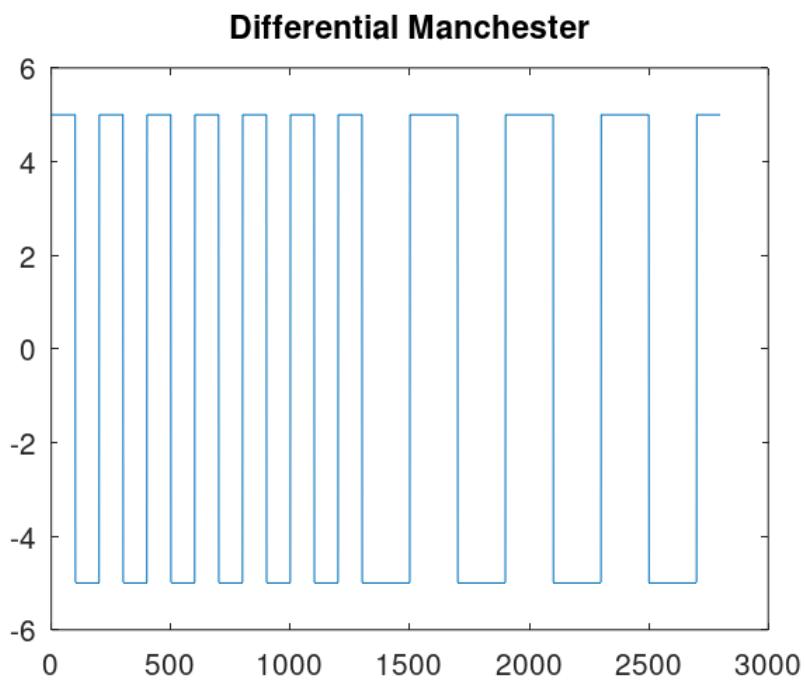


Рисунок 2.44: Дифференциальное манчестерское кодирование: есть самосинхронизация

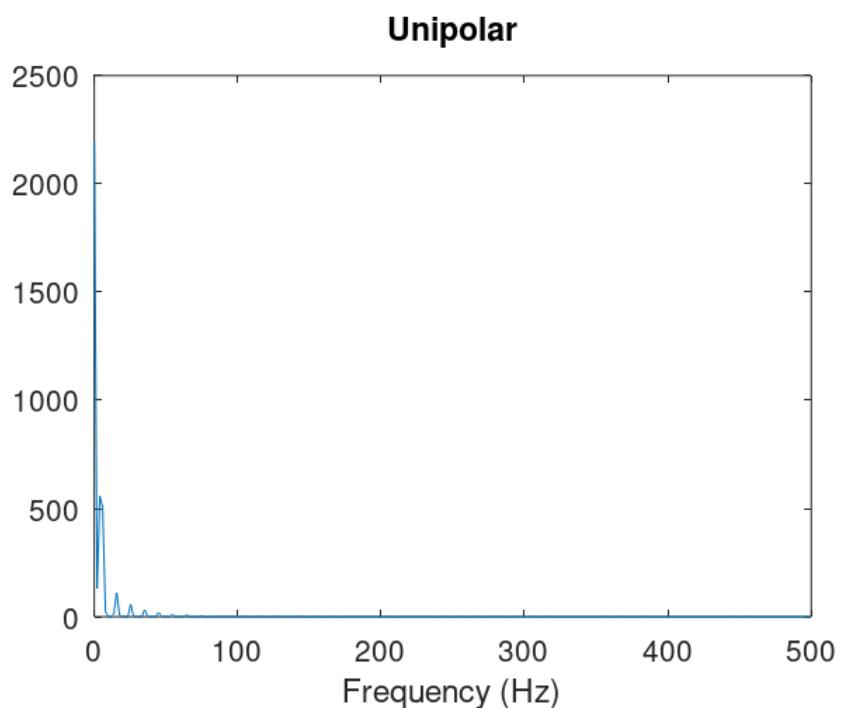


Рисунок 2.45: Униполярное кодирование: спектр сигнала

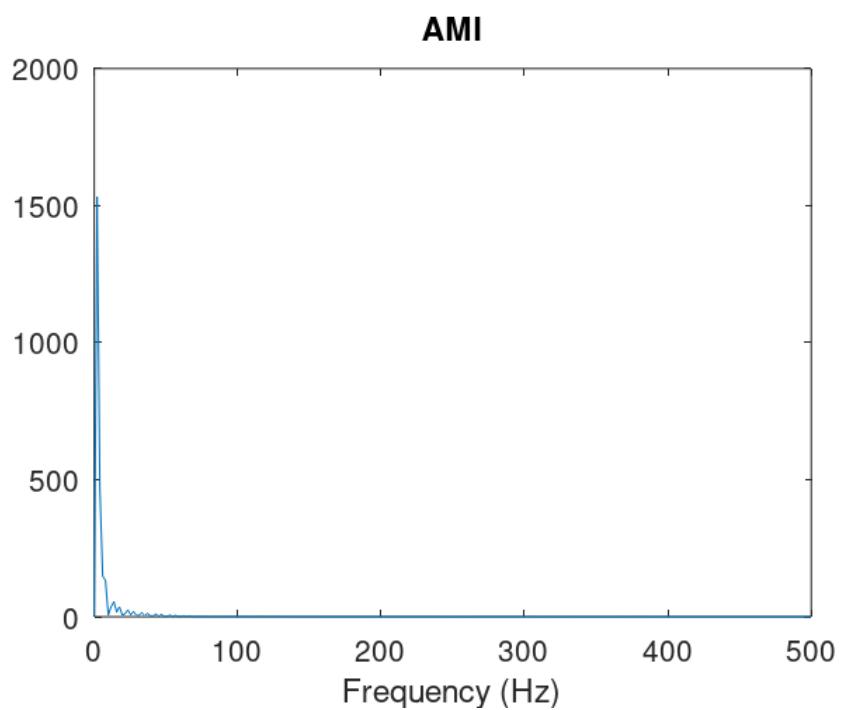


Рисунок 2.46: Кодирование AMI: спектр сигнала

Bipolar Non-Return to Zero

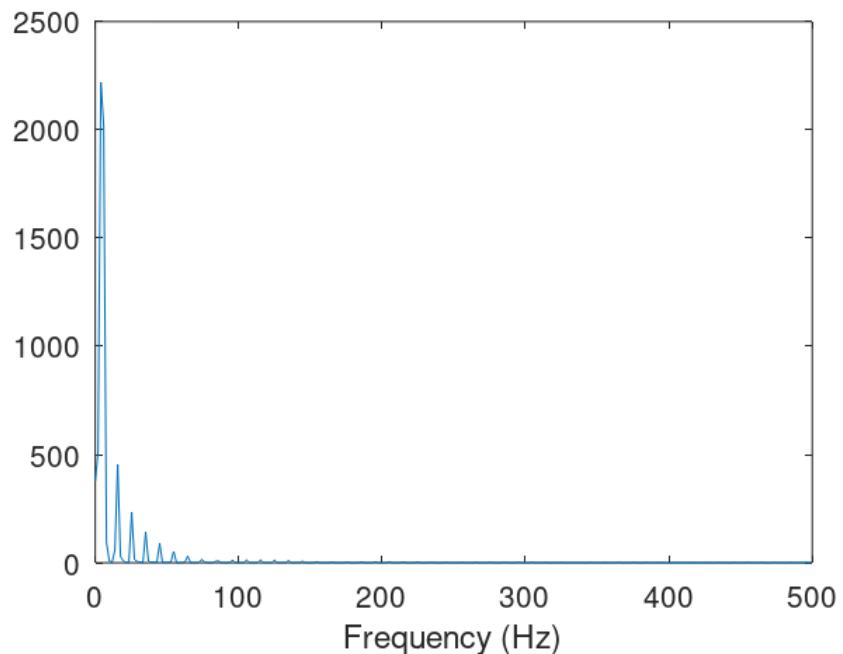


Рисунок 2.47: Кодирование NRZ: спектр сигнала

Bipolar Return to Zero

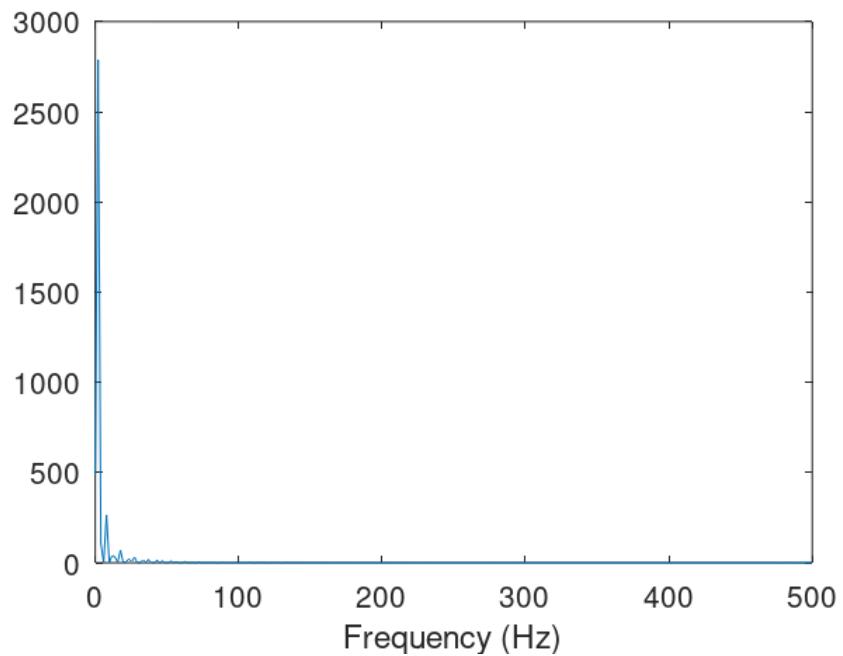


Рисунок 2.48: Кодирование RZ: спектр сигнала

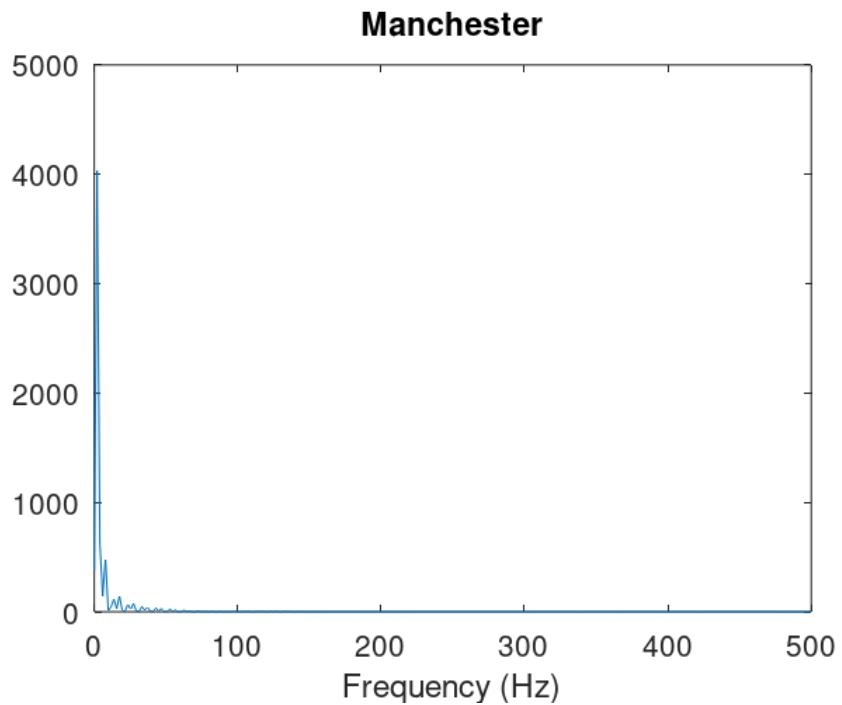


Рисунок 2.49: Манчестерское кодирование: спектр сигнала

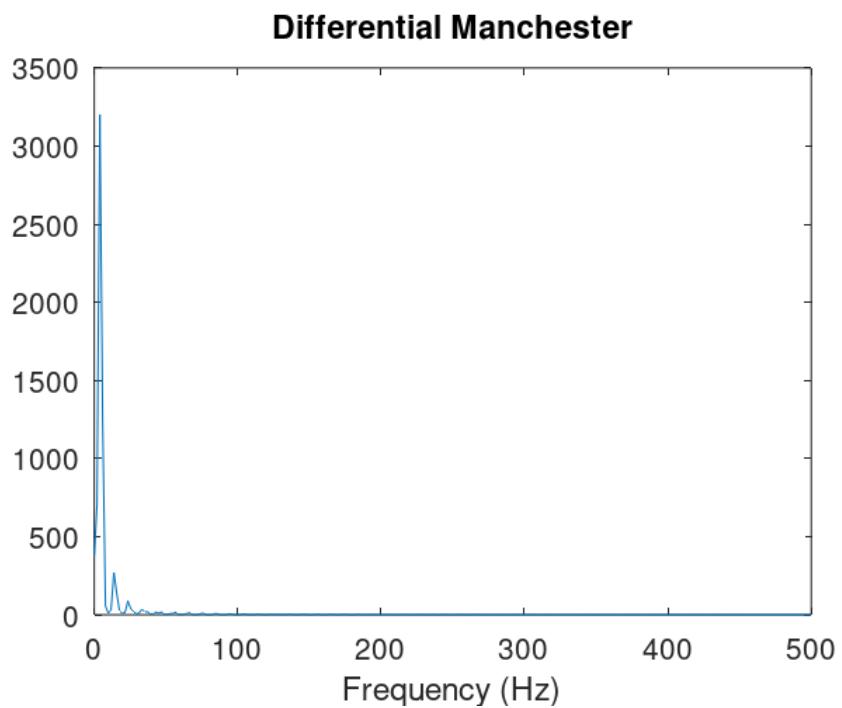


Рисунок 2.50: Дифференциальное манчестерское кодирование: спектр сигнала

3 Выводы

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