

Лабораторная работа № 1.

Методы кодирования и модуляция сигналов

Садова Д. А.

Российский университет дружбы народов, Москва, Россия

Информация

- Садова Диана Алексеевна
- студент бакалавриата
- Российский университет дружбы народов
- [113229118@pfur.ru]
- <https://DianaSadova.github.io/ru/>

Вводная часть

- Понять как взаимосвязаны сигнал и спектор.
- Разобраться в работе ОС Octave

- 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.

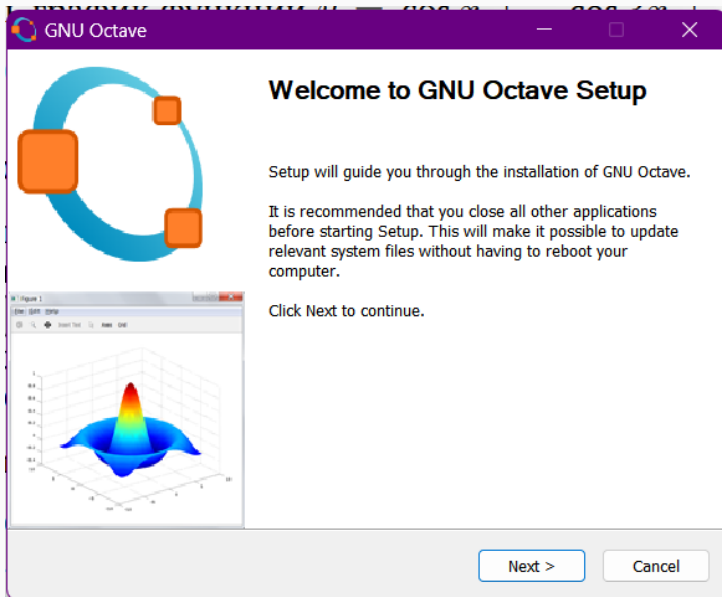
- Текст лабораторной работы № 1
- Интернет для исправления ошибок

- Построение графиков в Octave
- Разложение импульсного сигнала в частичный ряд Фурье
- Определение спектра и параметров сигнала
- Амплитудная модуляция
- Кодирование сигнала. Исследование свойства самосинхронизации сигнала

Построение графиков в Octave. 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`.

Порядок выполнения работы



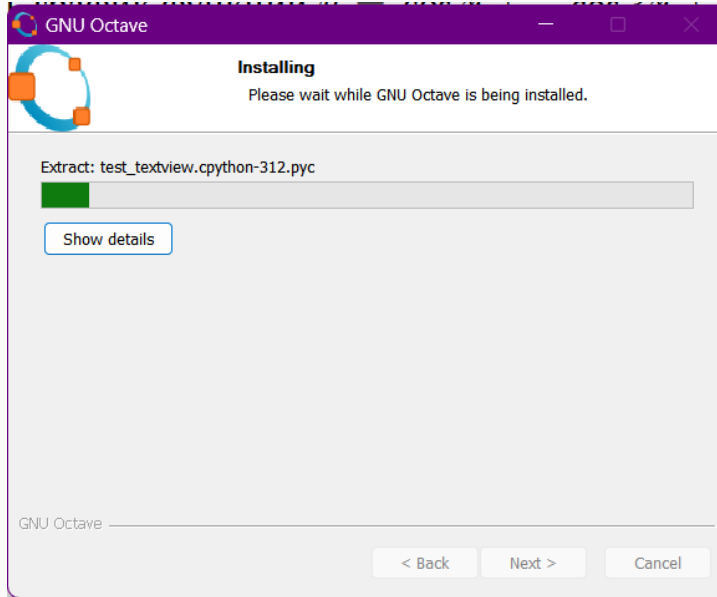


Рис. 2: Процесс установки

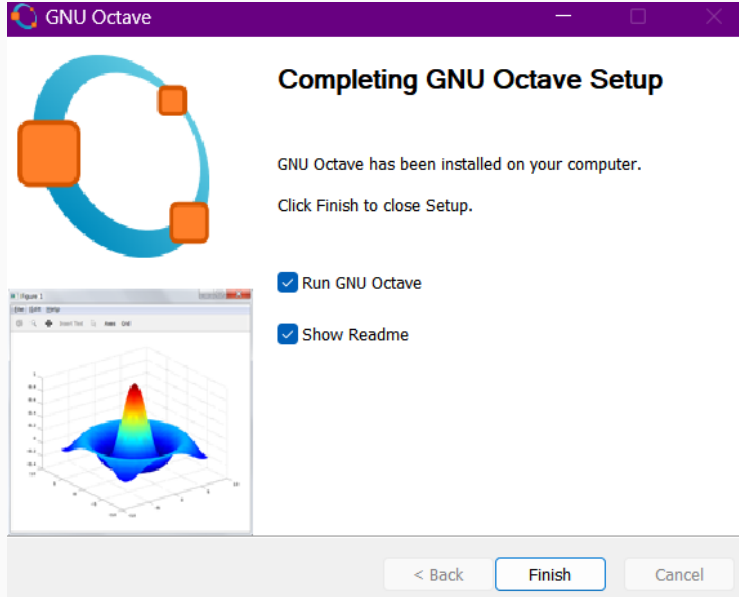


Рис. 3: Octave

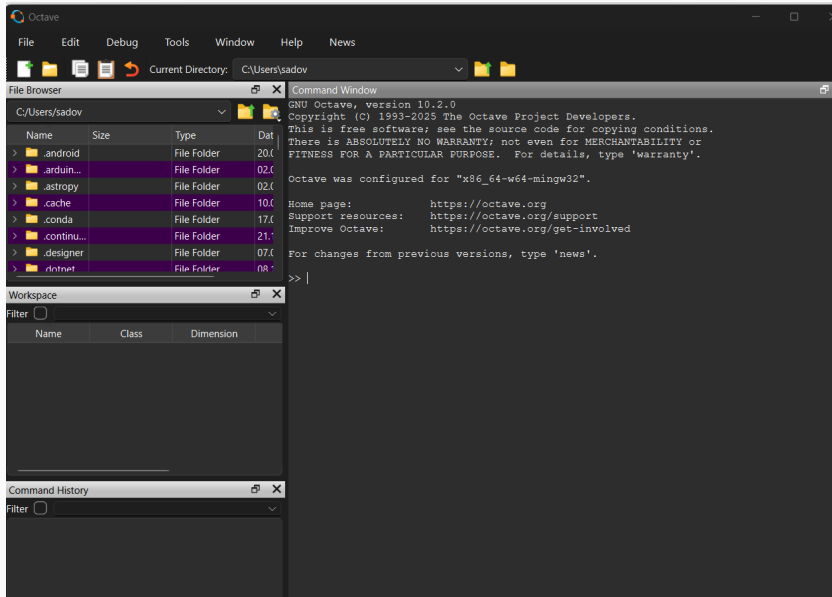


Рис. 4: ОС Octave с оконным интерфейсом

1. В окне редактора повторите следующий листинг по построению графика функции $y = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$ на интервале $[-10; 10]$:

```
<unnamed>
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
```

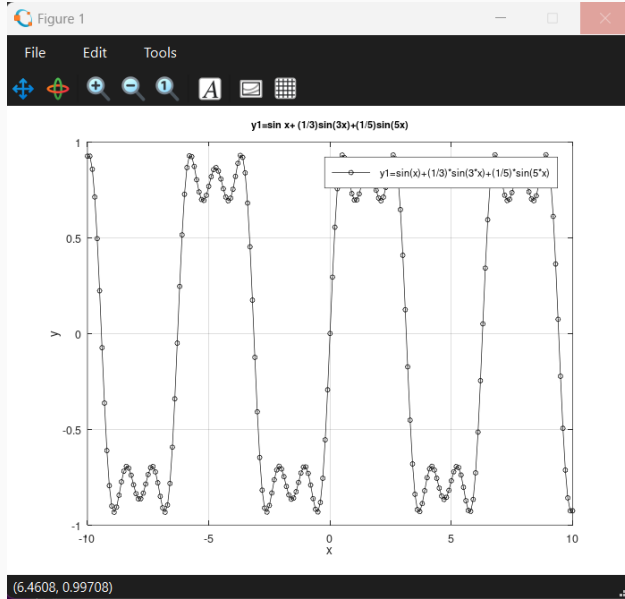


Рис. 6: График функции $y = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$

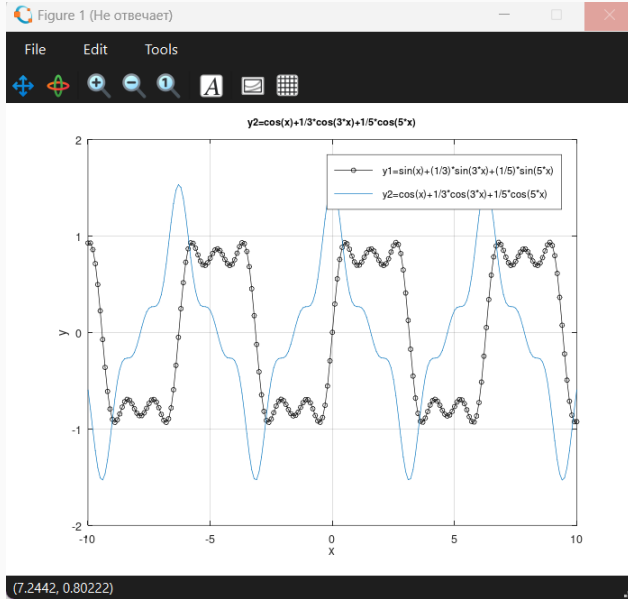


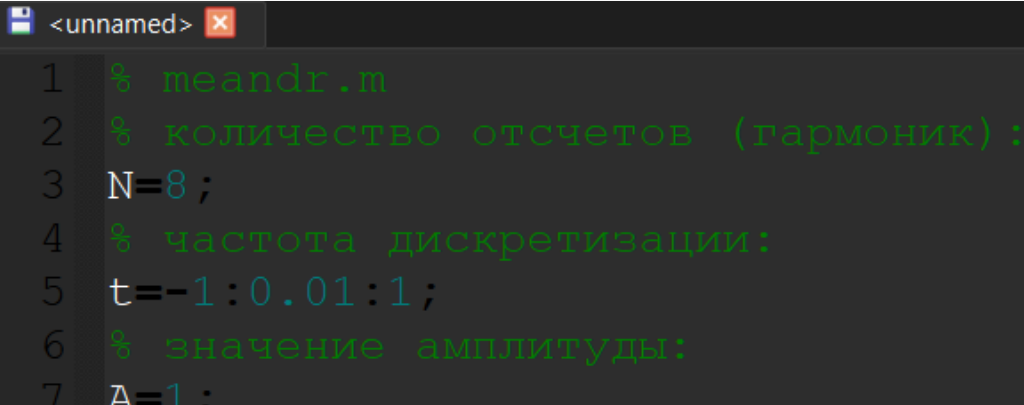
Рис. 7: График функций y_1 и y_2

Разложение импульсного сигнала в частичный ряд Фурье. Постановка задачи

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

Порядок выполнения работы

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



```
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 ;
```

```

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. 9: Код meandr.m

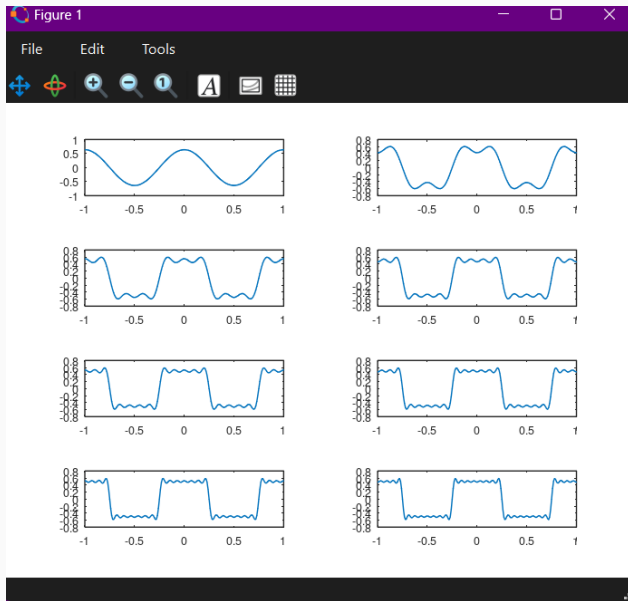
```

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
--

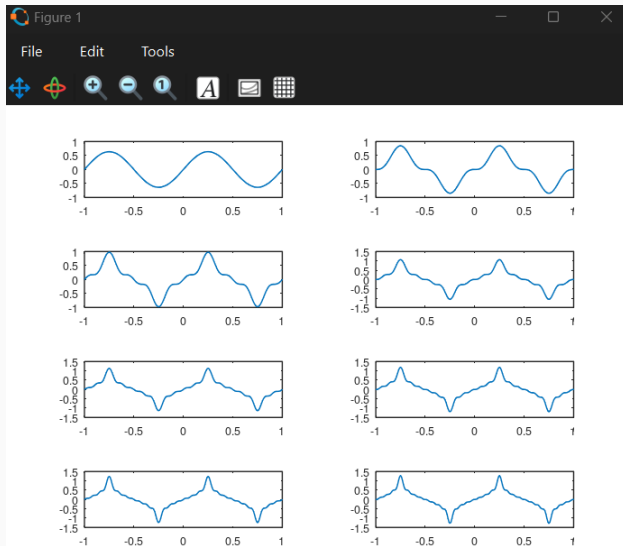
```

Рис. 10: Код meandr.m

5. Экспортируйте полученный график в файл в формате .png.



6. Скорректируйте код для реализации меандра через синусы. Получите соответствующие графики.



Определение спектра и параметров сигнала. 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?

Порядок выполнения работы

1. В вашем рабочем каталоге создайте каталог spectre1 и в нём новый сценарий с именем, spectre.m.

```
% 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
```

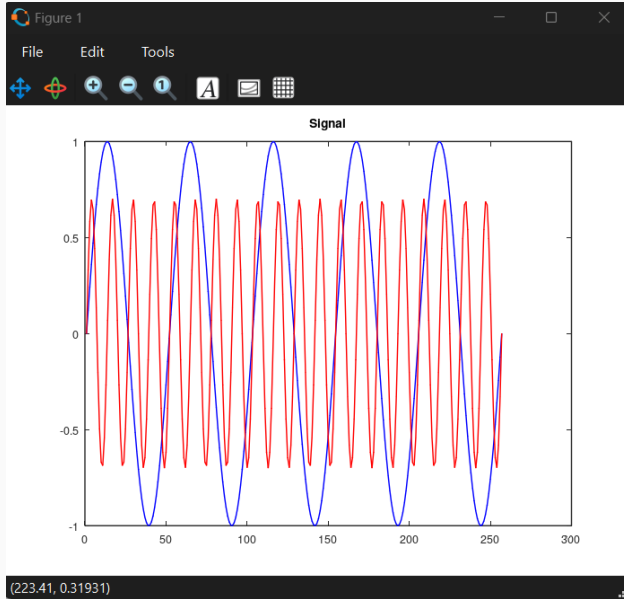



Рис. 14: Два синусоидальных сигнала разной частоты

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:

```
% 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
```

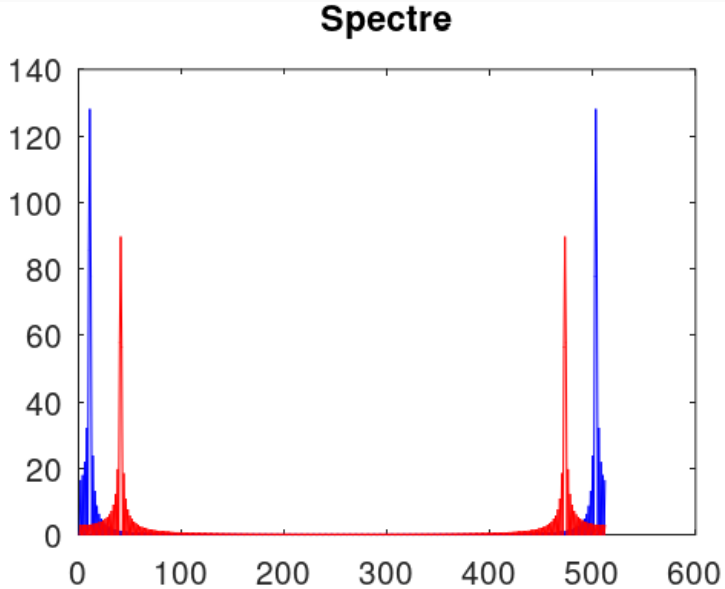


Рис. 17: График спектров синусоидальных сигналов

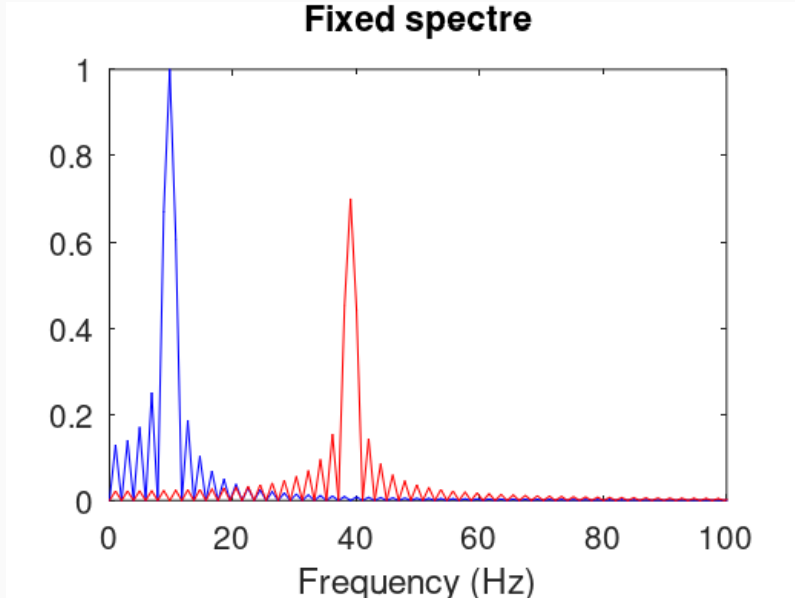


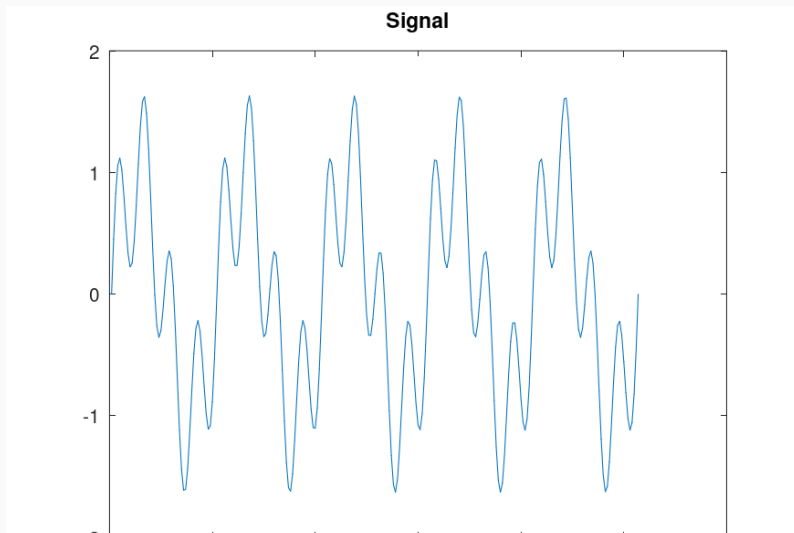
Рис. 18: Исправленный график спектров синусоидальных сигналов

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

```
% spectr_sum/spectre_sum.m
% Создание каталогов signal и spectre для размещения графиков:
mkdir 'signal';
mkdir 'spectre';
% Длина сигнала (s):
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;
% Число отсчетов
fd2 = fd/2;
% Сумма двух сигналов (синусоиды разной частоты):
% Вектор 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';
% Выходной спектр:
% Амплитуды преобразованных типов сигнала:
spectre = fft(signal,fd);
% Частота 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)');
```

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



Spectre

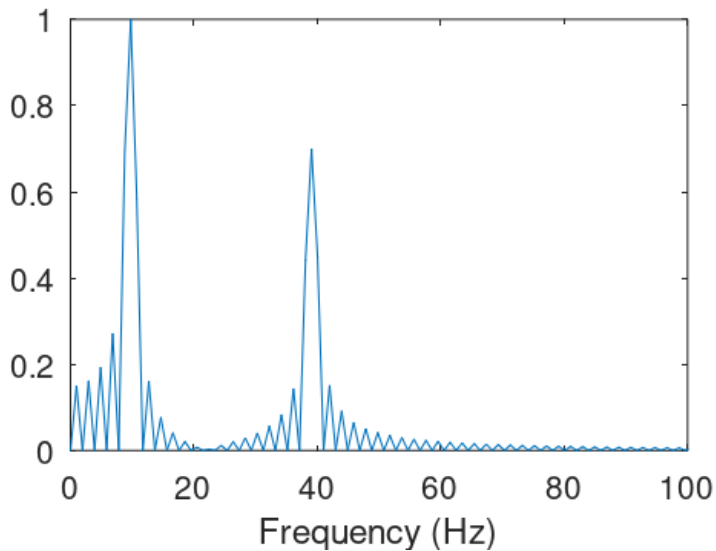


Рис. 21: Спектр суммарного сигнала

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)

Амплитудная модуляция. Постановка задачи

Продemonстрировать принципы модуляции сигнала на примере аналоговой амплитудной модуляции

Порядок выполнения работы

1. В вашем рабочем каталоге создайте каталог modulation и в нём новый сценарий с именем am.m.

```
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 % Шагер сигнала
15 fd2 = fd/2;
16 % Построение графиков двух сигналов (синусоиды)
17 % разной частоты
18 % Масштаб 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 графиков:
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 % Длина частот:
35 f = 1000*(0:fd2)/(2*fd);
```

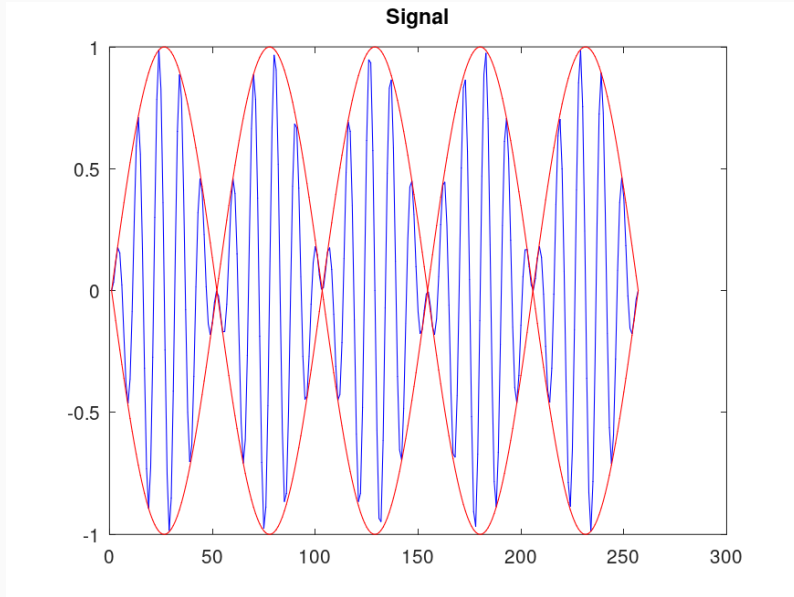


Рис. 23: Сигнал и огибающая при амплитудной модуляции

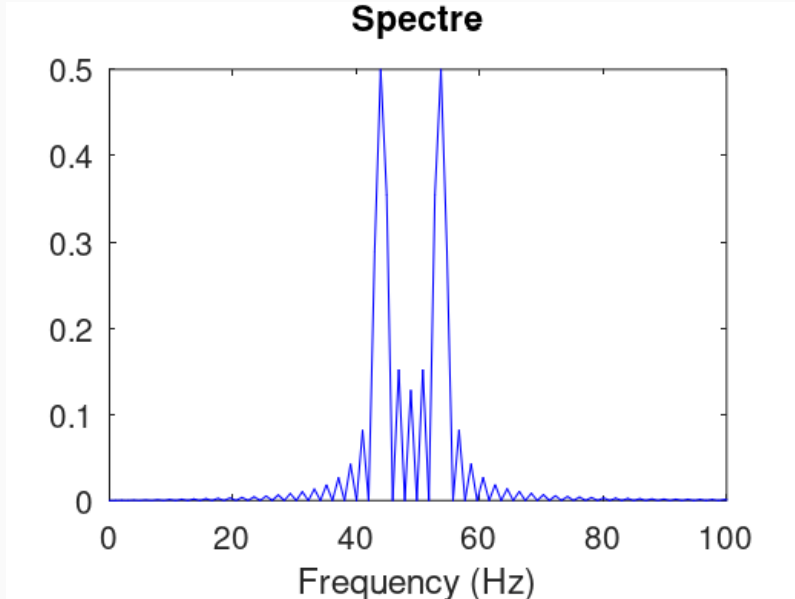


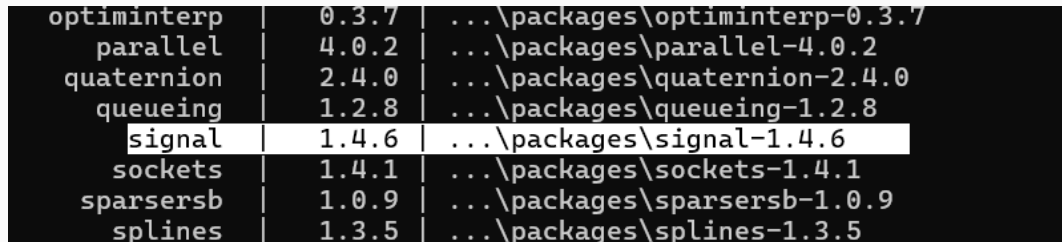
Рис. 24: Спектр сигнала при амплитудной модуляции

Кодирование сигнала. Исследование свойства самосинхронизации сигнала. Постановка задачи

По заданных битовых последовательностей требуется получить кодированные сигналы для нескольких кодов, проверить свойства самосинхронизируемости кодов, получить спектры.

Порядок выполнения работы

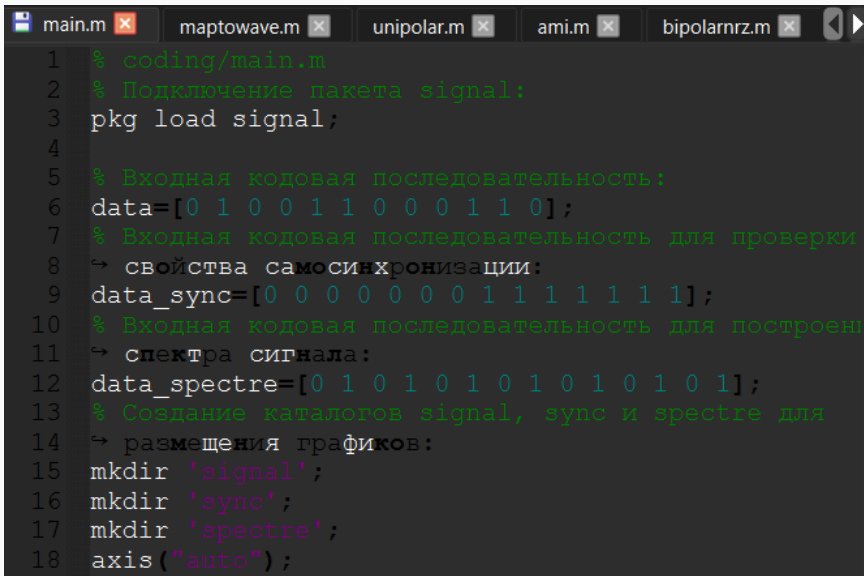
1. В вашем рабочем каталоге создайте каталог coding и в нём файлы main.m, maptowave.m, unipolar.m, ami.m, bipolarnrz.m, bipolarrrz.m, manchester.m, diffmanc.m, calcspectre.m.
2. В окне интерпретатора команд проверьте, установлен ли у вас пакет расширений signal:



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. 25: Окно терминала. Проверяем наличие signal

3. В файле main.m подключите пакет signal и задайте входные кодовые последовательности:



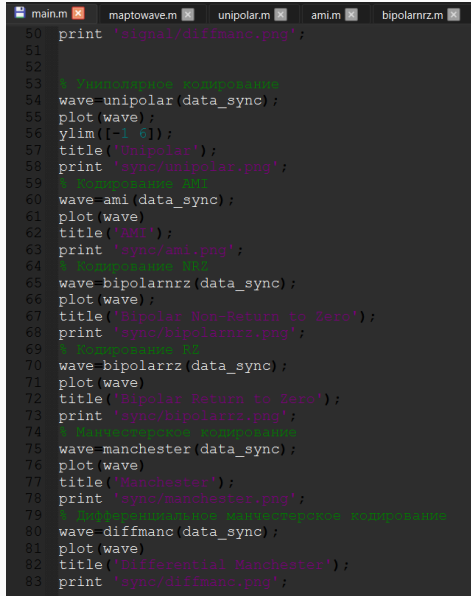
```
1 % coding/main.m
2 % Подключение пакета signal:
3 pkg load signal;
4
5 % Входная кодовая последовательность:
6 data=[0 1 0 0 1 1 0 0 0 1 1 0];
7 % Входная кодовая последовательность для проверки
8   ↳ свойства самосинхронизации:
9 data_sync=[0 0 0 0 0 0 0 1 1 1 1 1 1];
10 % Входная кодовая последовательность для построения
11   ↳ спектра сигнала:
12 data_spectre=[0 1 0 1 0 1 0 1 0 1 0 1 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");
```

```

main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m x
9 data_sync=[0 0 0 0 0 0 0 1 1 1 1 1 1];
10 % Видная 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 aml
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. 27: Код main.m



```

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=bipolarrz(data_sync);
71 plot(wave);
72 title('Bipolar Return to Zero');
73 print 'sync/bipolarrz.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';

```

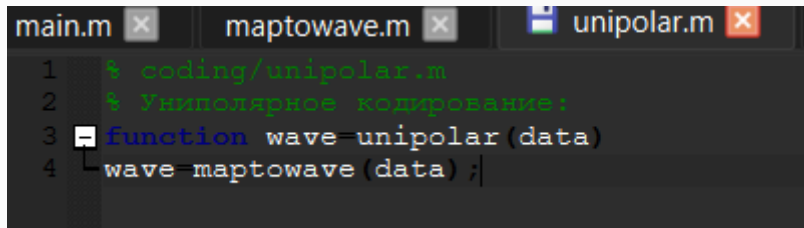
Рис. 28: Код main.m

```
main.m x mptowave.m x unipolar.m x ami.m x bipolarnrz.m
82 title('Differential Manchester');
83 print 'sync/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. 29: Код main.m

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

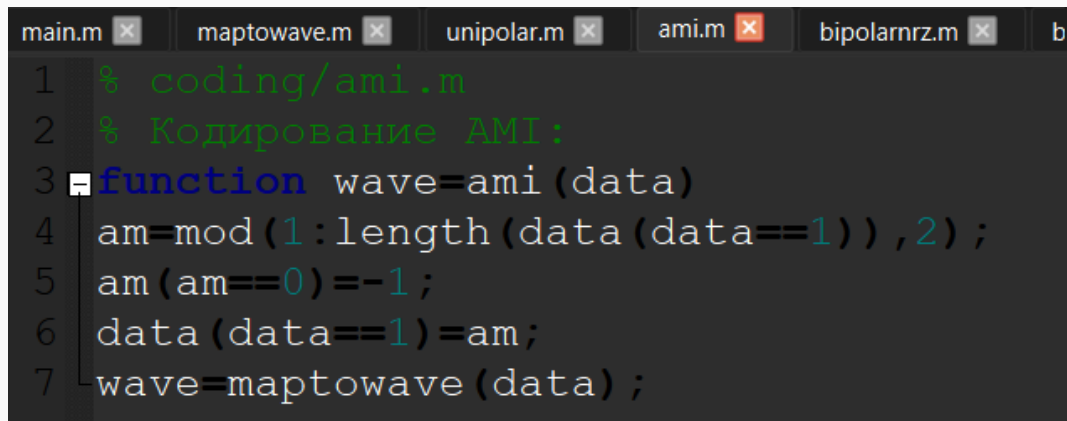
Униполярное кодирование:



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

Рис. 31: Код unipolar.m

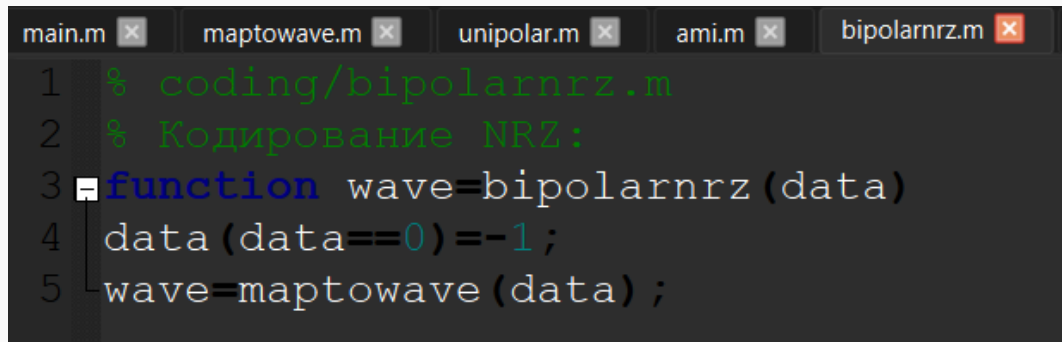
Кодирование AMI:



```
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. 32: Код ami.m

Кодирование NRZ:

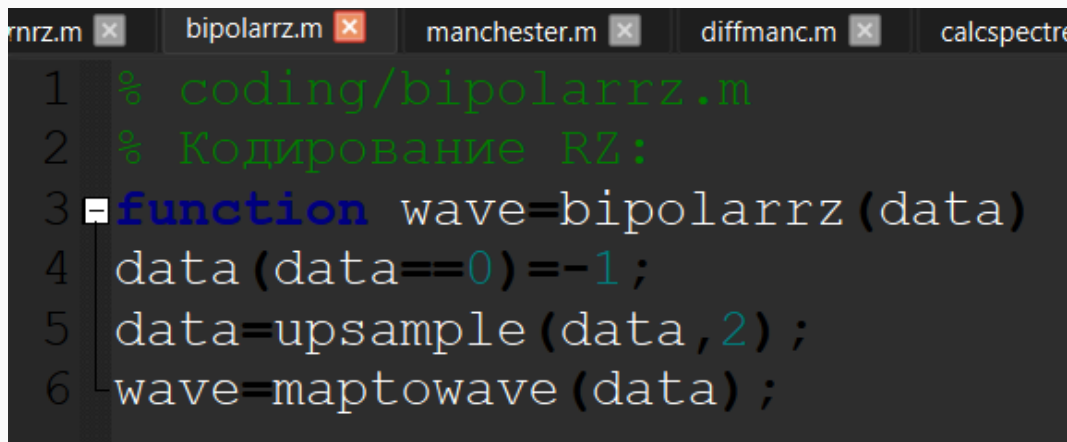


The image shows a MATLAB script editor with five tabs: main.m, maptowave.m, unipolar.m, ami.m, and bipolarnrz.m. The active tab is bipolarnrz.m, which contains the following code:

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

Рис. 33: Код bipolarnrz.m

Кодирование RZ:

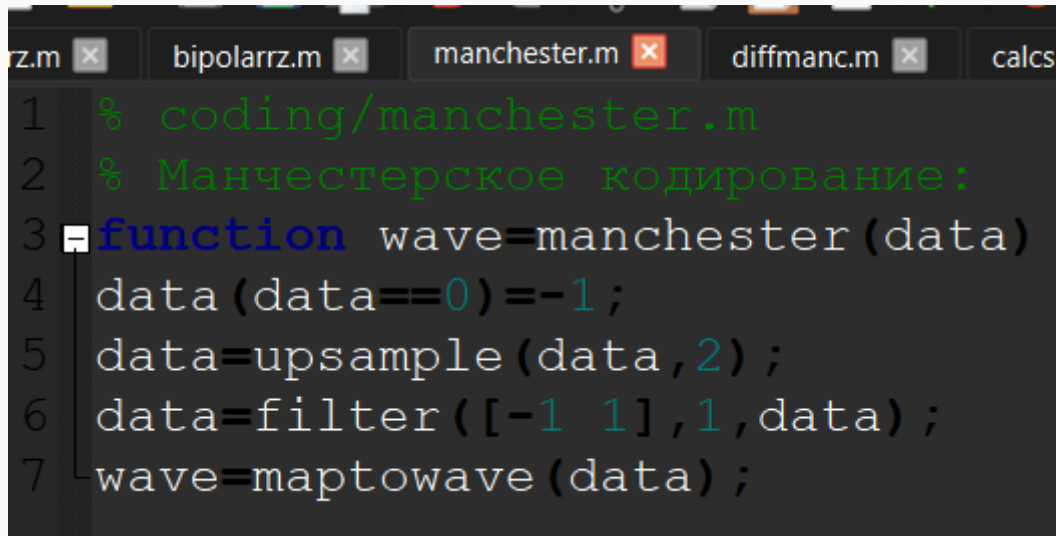


The image shows a MATLAB code editor window with several tabs at the top: 'rnrz.m', 'bipolarrz.m' (active), 'manchester.m', 'diffmanc.m', and 'calcspectre'. The code in the active tab is as follows:

```
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);
```

Рис. 34: Код bipolarrrz.m

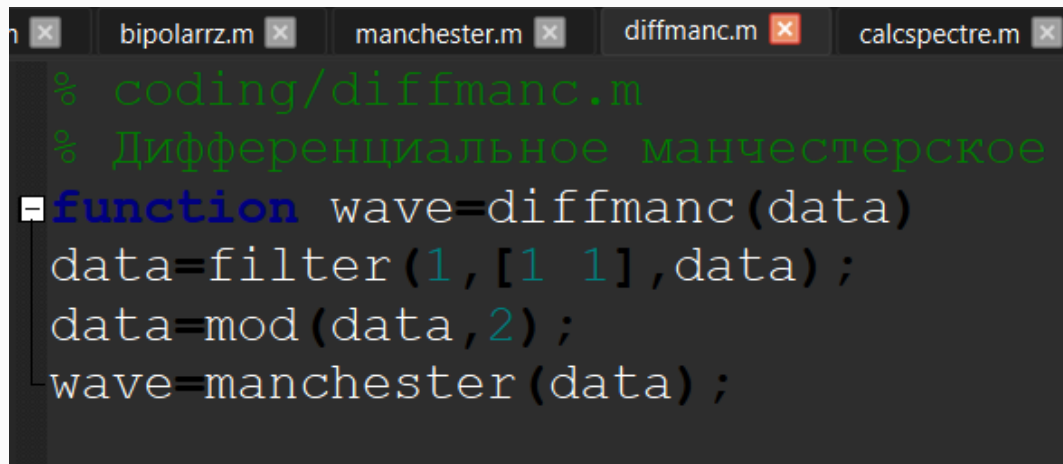
Манчестерское кодирование:



```
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. 35: Код manchester.m

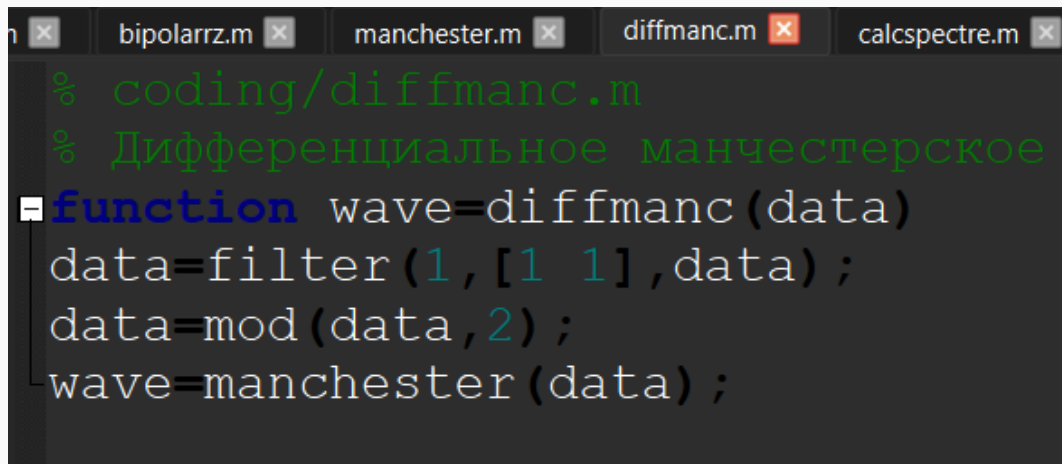
Дифференциальное манчестерское кодирование:



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

Рис. 36: Код diffmanc.m

6. В файле calcspectre.m пропишите функцию построения спектра сигнала:

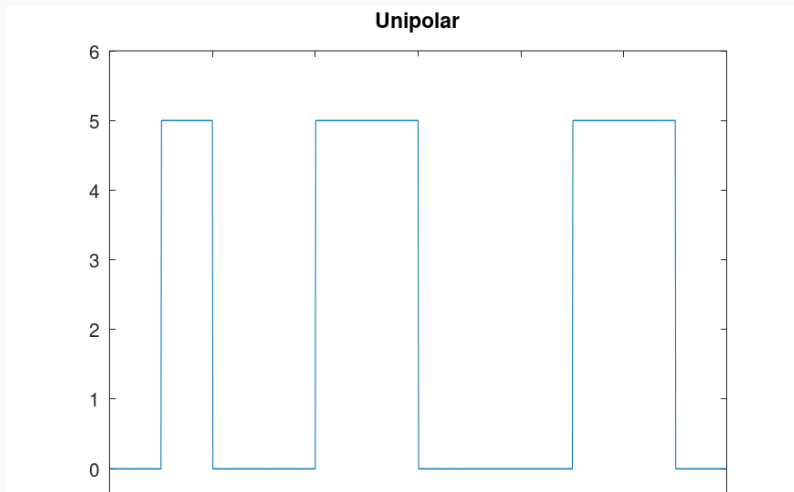


The image shows a MATLAB script editor window with several tabs at the top: 'bipolarrz.m', 'manchester.m', 'diffmanc.m', and 'calcspectre.m'. The 'calcspectre.m' tab is active. The code in the editor is as follows:

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

Рис. 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.



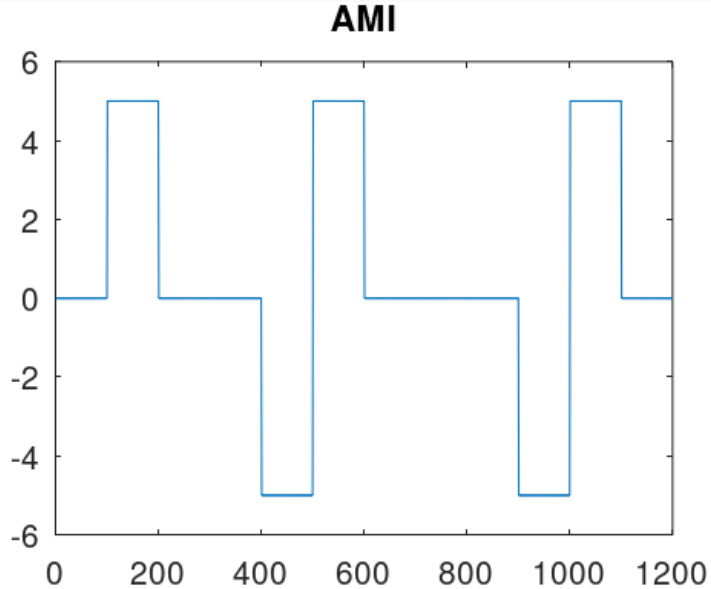


Рис. 39: Кодирование AMI

Bipolar Non-Return to Zero

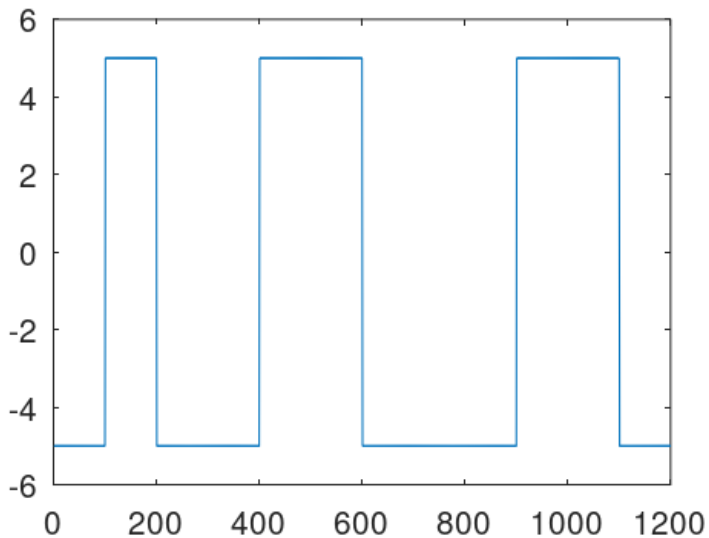


Рис. 40: Кодирование NRZ

Bipolar Return to Zero

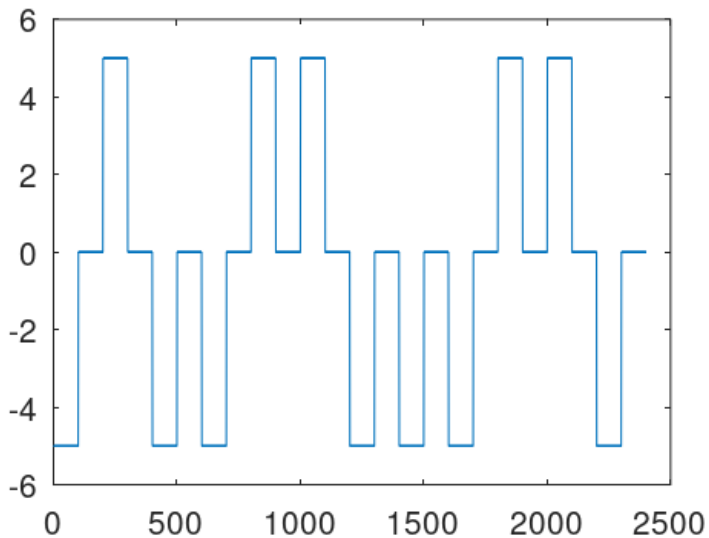


Рис. 41: Кодирование RZ

Manchester

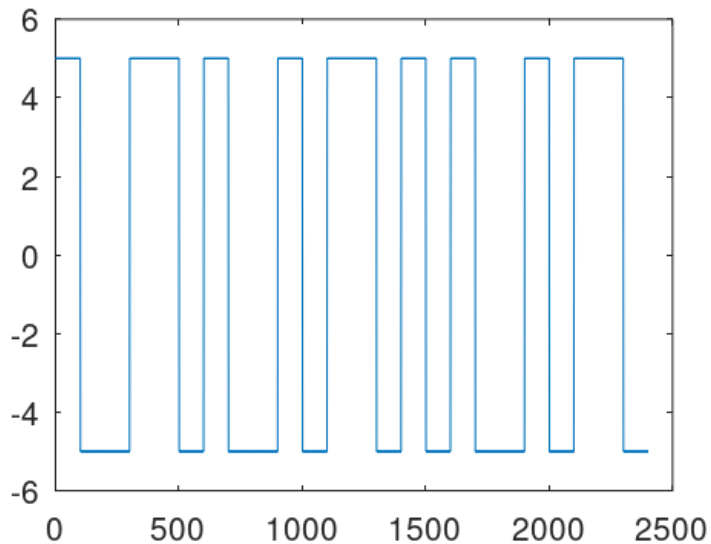


Рис. 42: Манчестерское кодирование

Differential Manchester

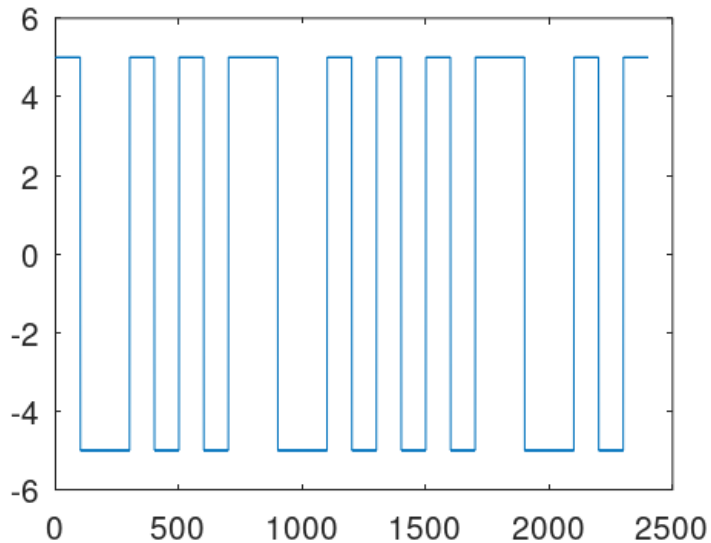


Рис. 43: Дифференциальное манчестерское кодирование

Unipolar

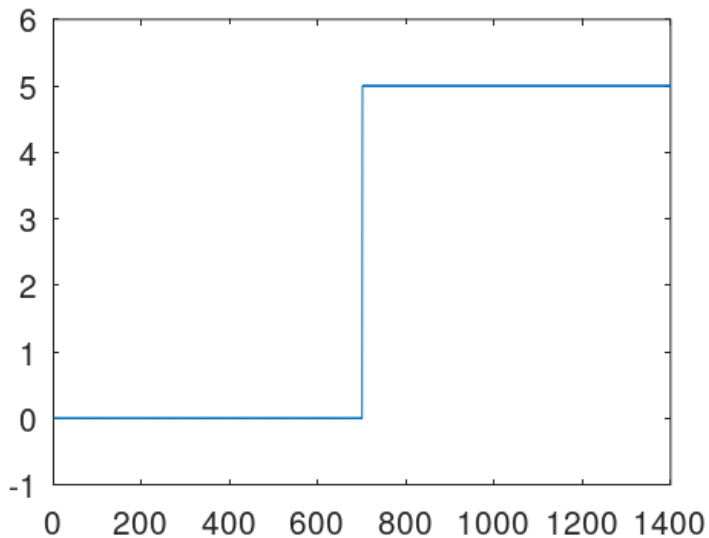


Рис. 44: Униполярное кодирование: нет самосинхронизации

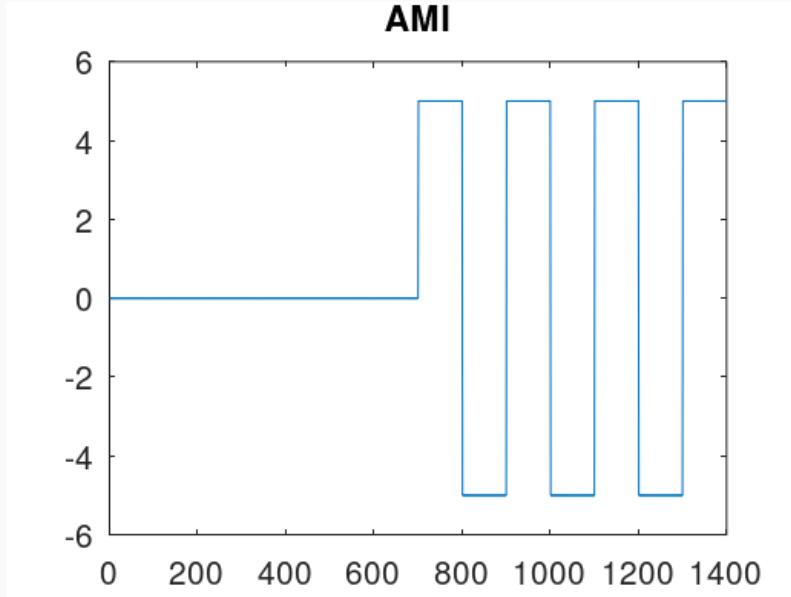


Рис. 45: Кодирование AMI: самосинхронизация при наличии сигнала

Bipolar Non-Return to Zero

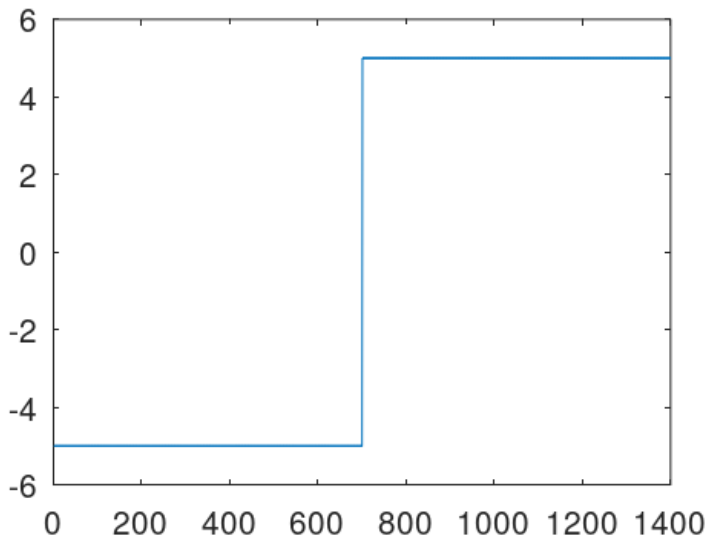


Рис. 46: Кодирование NRZ: нет самосинхронизации

Bipolar Return to Zero

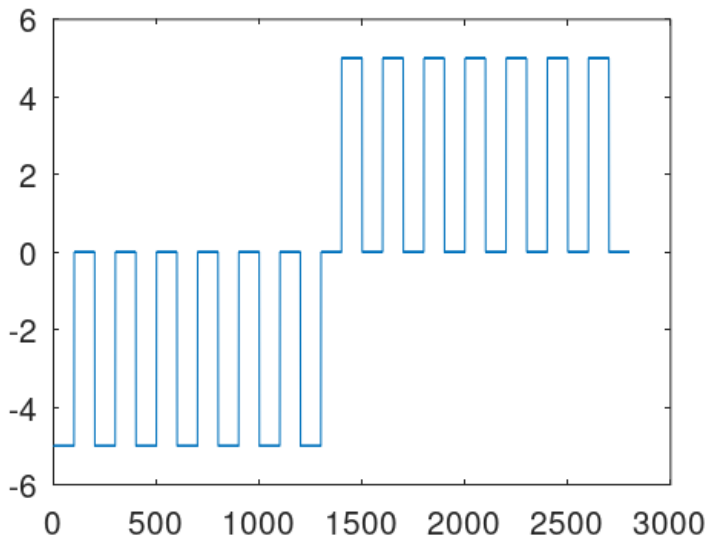


Рис. 47: Кодирование RZ: есть самосинхронизация

Manchester

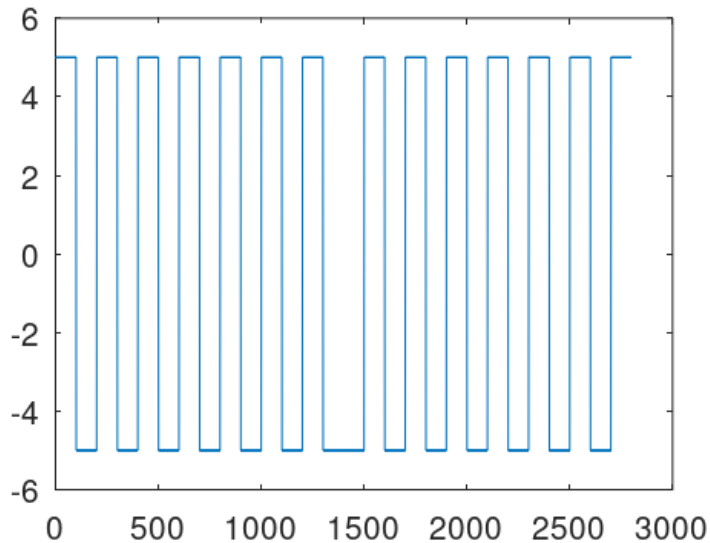


Рис. 48: Манчестерское кодирование: есть самосинхронизация

Differential Manchester

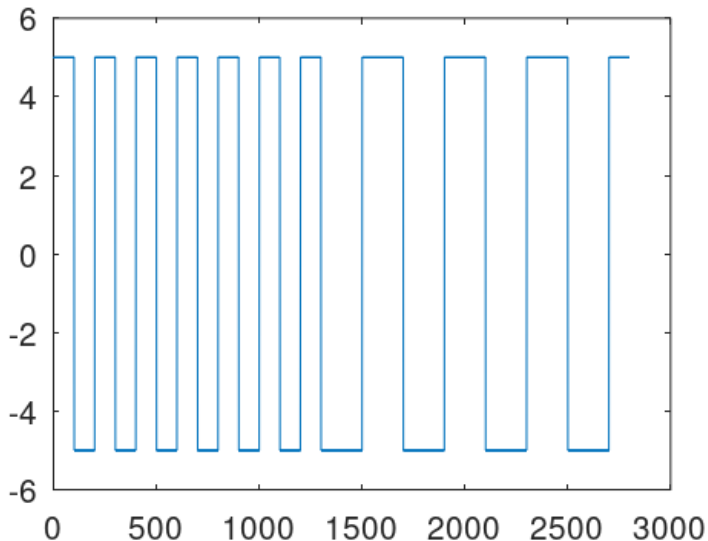


Рис. 49: Дифференциальное манчестерское кодирование: есть самосинхронизация ^{59/66}

Unipolar

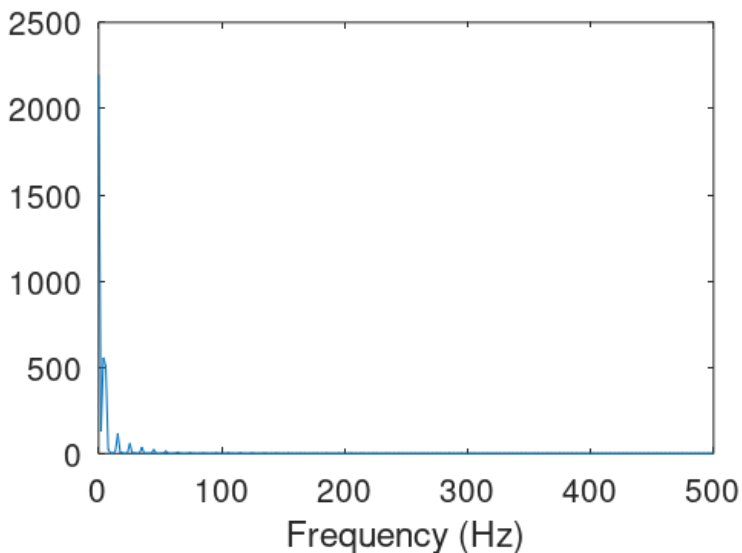


Рис. 50: Униполярное кодирование: спектр сигнала

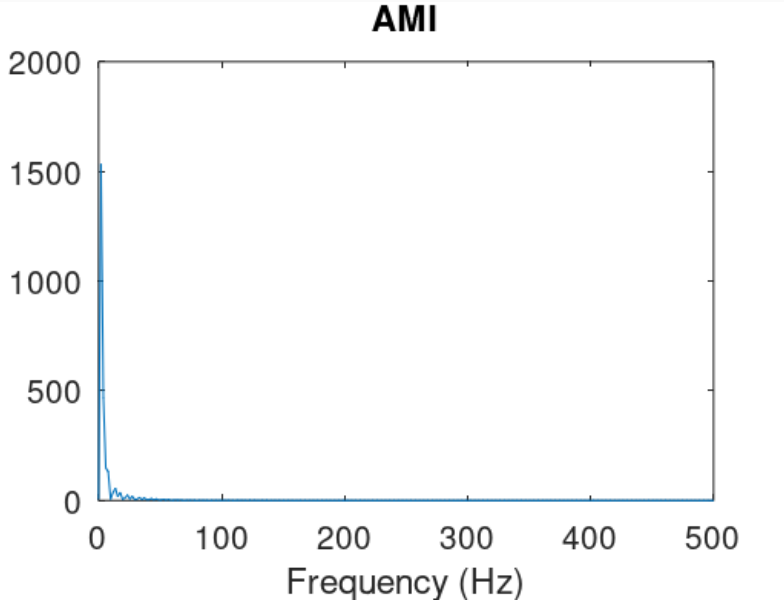


Рис. 51: Кодирование AMI: спектр сигнала

Bipolar Non-Return to Zero

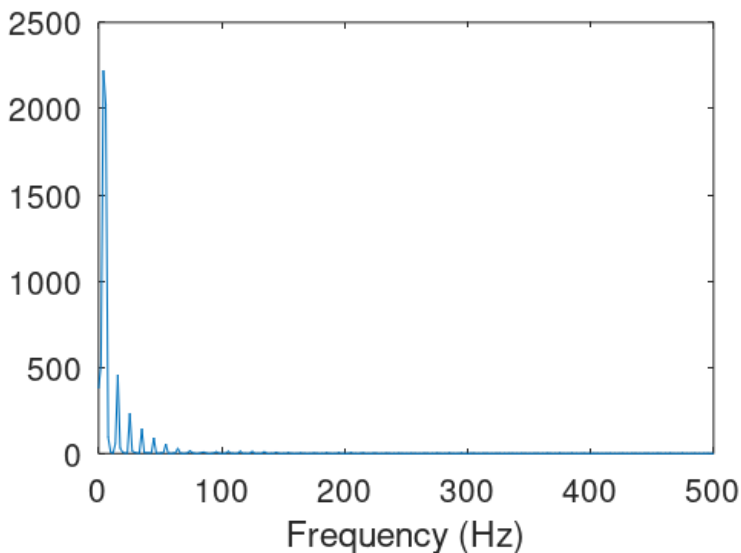


Рис. 52: Кодирование NRZ: спектр сигнала

Bipolar Return to Zero

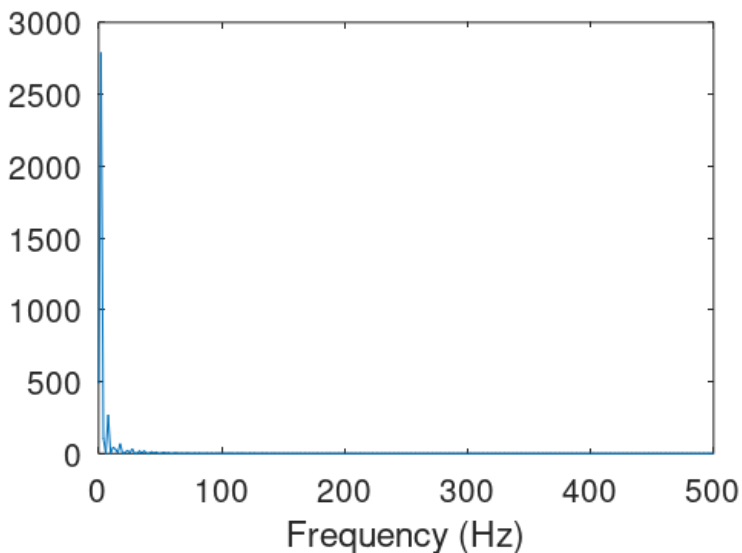


Рис. 53: Кодирование RZ: спектр сигнала

Manchester

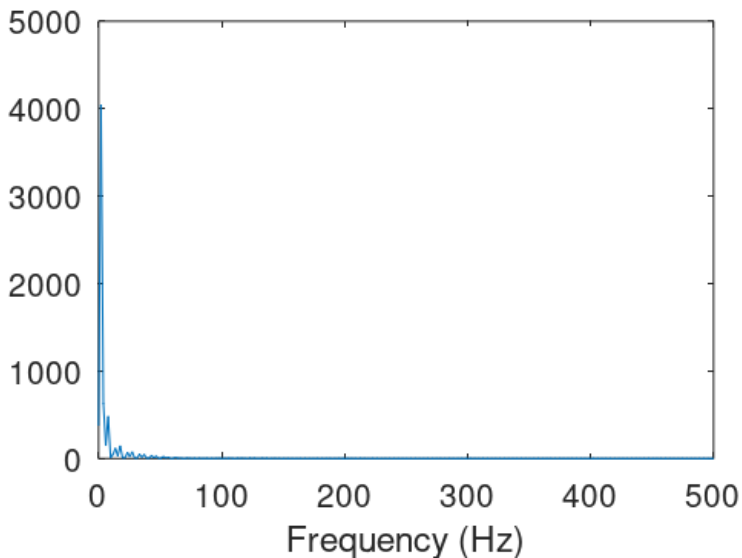


Рис. 54: Манчестерское кодирование: спектр сигнала

Differential Manchester

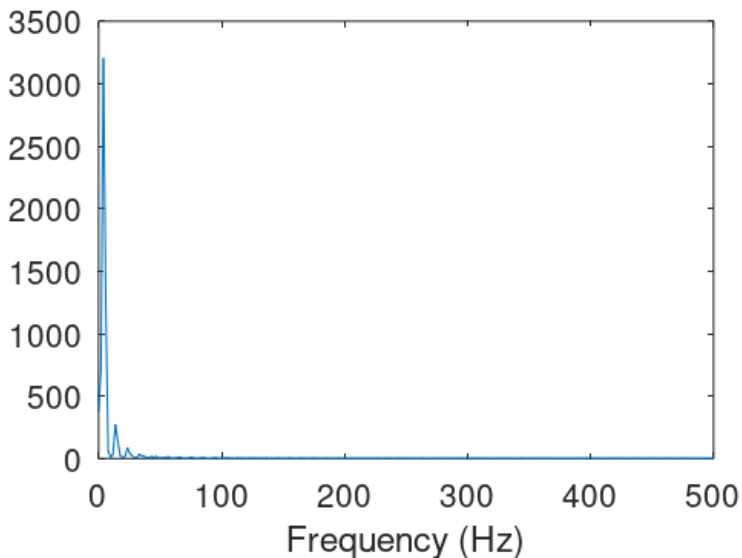


Рис. 55: Дифференциальное манчестерское кодирование: спектр сигнала

- 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.