Отчет по лабораторной работе № 1. Методы кодирования и модуляция сигналов

Данила Стариков
НПИбд-02-22

14 сентября 2024 г.

Содержание

1 Цель работы

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

2 Выполнение работы

2.1 Построение графиков в Octave

1. Запустили Octave в режиме графического интерфейса.

2. Создали новый файл `plot_sin.m`. В нем повторили листинг, который строит график функции $y = \sin x + \frac{1}{3}\sin 3x + \frac{1}{5}\sin 5x$ на интервале $[-10; 10]$:

```
% 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");
```

3. При запуске программы вывели график запрограммированной функции, который был сохранен в файл `plotsin.png` 1:

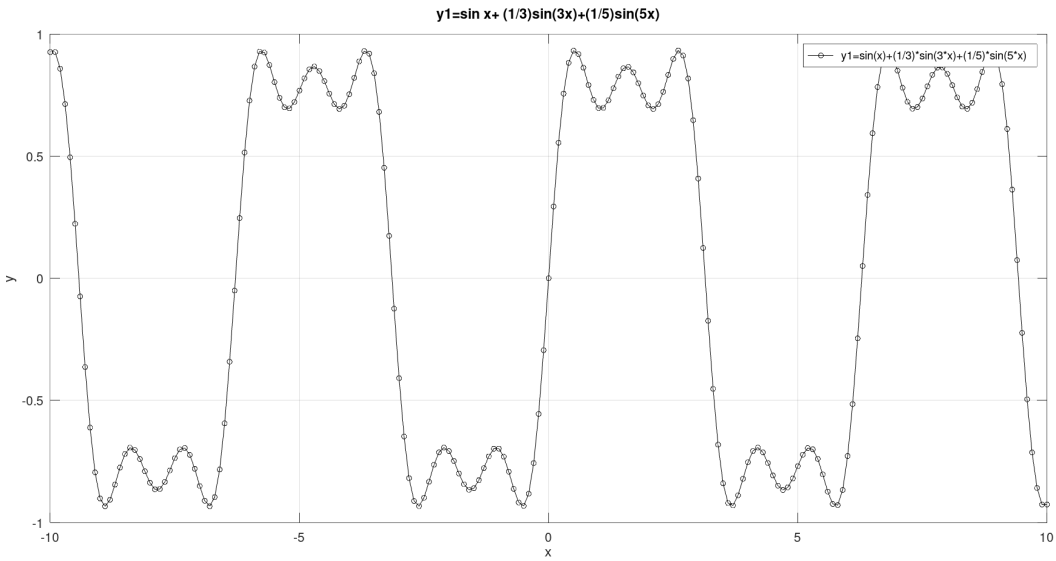

Рис. 1: График функции $y = \sin x + \frac{1}{3}\sin 3x + \frac{1}{5}\sin 5x$ на интервале $[-10; 10]$

4. В новом файле `plot_sin_cos.m` реализовали программу, которая строит на одном графике две линии, соответствующие функциям $y1 = \sin x + \frac{1}{3}\sin 3x + \frac{1}{5}\sin 5x, y2 = \cos x + \frac{1}{3}\cos `3x + \frac{1}{5}\cos 5x$:

```
% Формирование массива x:
x=-10:0.1:10;
% Формирование массива y.
y1=sin(x)+1/3*sin(3*x)+1/5*sin(5*x);
y2=cos(x)+1/3*cos(3*x)+1/5*cos(5*x);
% Построение графика функции:
plot(x,y1,"-ok; y1=sin(x)+(1/3)*sin(3*x)+(1/5)*sin(5*x);","markersize",4,
 ↪  x, y2, "; y2=cos(x)+(1/3)*cos(3*x)+(1/5)*cos(5*x);")
% 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); y2=cos
 ↪  x+(1/3)cos(3x)+(1/5)cos(5x)');
% Экспорт рисунка в файл .eps:
print ("plot-sincos.eps", "-mono", "-FArial:16", "-deps")
% Экспорт рисунка в файл .png:
print("plot-sincos.png");
```

5. В результате выполнения скрипта получен график, изображенный на Рисунке 2:

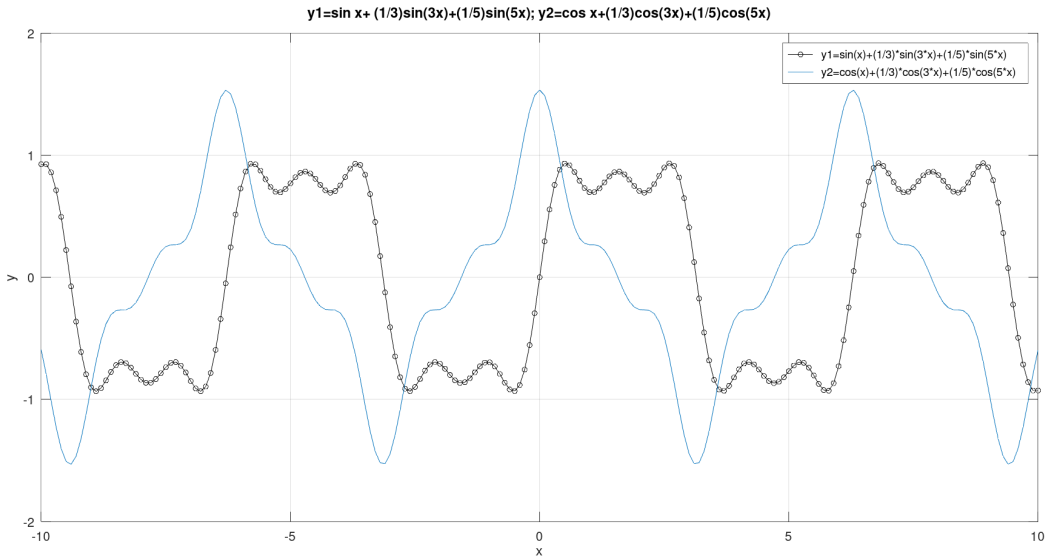

Рис. 2: График функций $y1 = \sin x + \frac{1}{3}\sin 3x + \frac{1}{5}\sin 5x, y2 = \cos x + \frac{1}{3}\cos `3x + \frac{1}{5}\cos 5x$ на интервале $[-10; 10]$

2.2 Разложение импульсного сигнала в частичный ряд Фурье

1. Создали новый файл `meandr_sin.m`, в котором реализовали программу, которая строит графики меандра с разными числом гармоник, используя синусы (Рисунок 3). Листинг программы представлен ниже:

```
% 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=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,:))
  xticks(-1:.5:1)
  yticks(-1:.5:1)
end
% Экспорт рисунка в файл .png:
print("plot-meandr-sin.png");
```

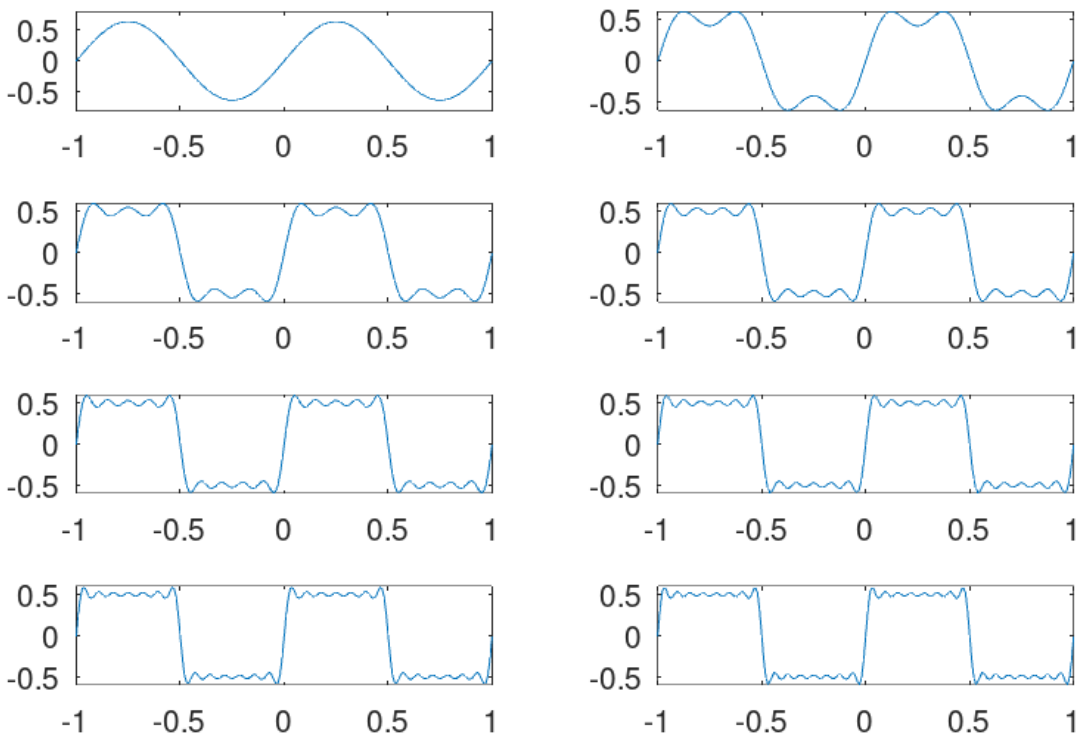

Рис. 3: Графики меандра, с различным числом гармоник (синусы).

2. Создали новый файл `meandr_cos.m`, в котором реализовали программу, которая строит графики меандра с разными числом гармоник, используя косинусы (Рисунок 4). Листинг программы представлен ниже:

```
% 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,:))
  xticks(-1:.5:1)
  yticks(-1:.5:1)
end

% Экспорт рисунка в файл .png:
print("plot-meandr-cos.png");
```

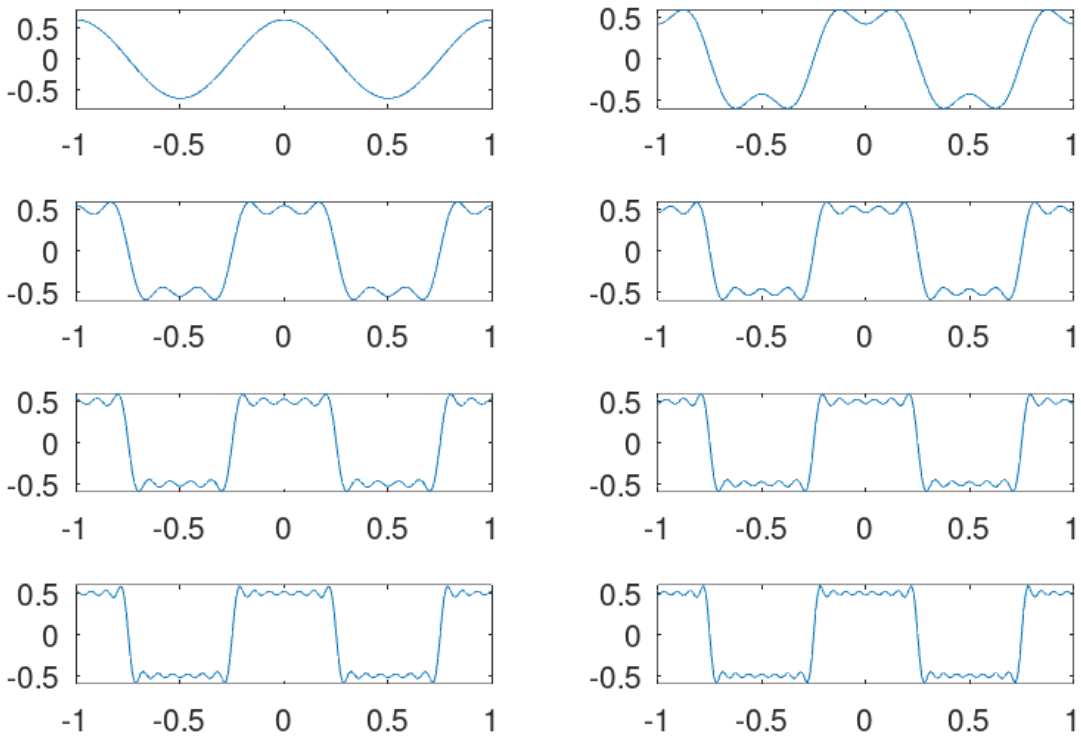

Рис. 4: графики меандра, с различным числом гармоник (косинусы).

2.3 Определение спектра и параметров сигнала

1. Создали каталог `spectre1` и в нем файл `spectre.m`. В файле реализовали программу, которая строит график с двумя сигналами разной частоты, затем с помощью быстрого преобразования Фурье находит спектры этих сигналов (Рисунки 5-7)

```
% 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';

% 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';
```

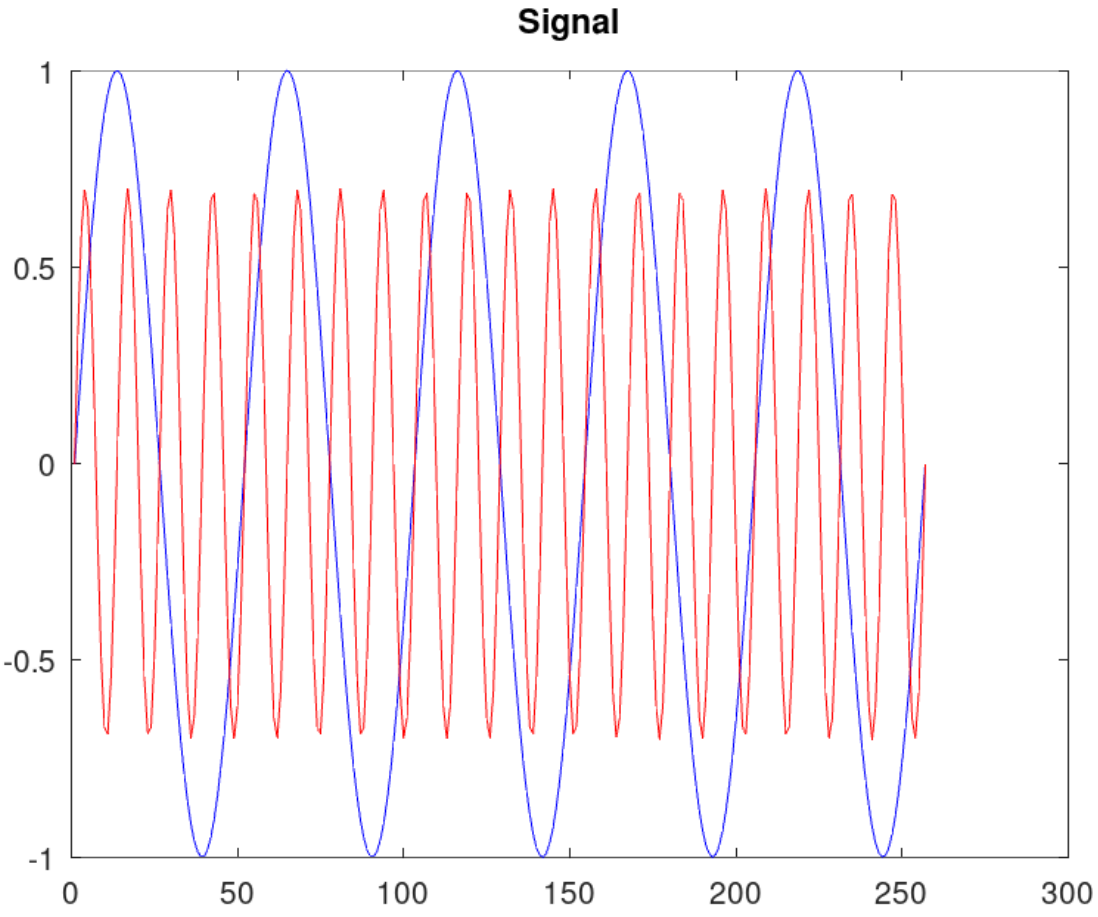

Рис. 5: Два синусоидальных сигнала разной частоты.

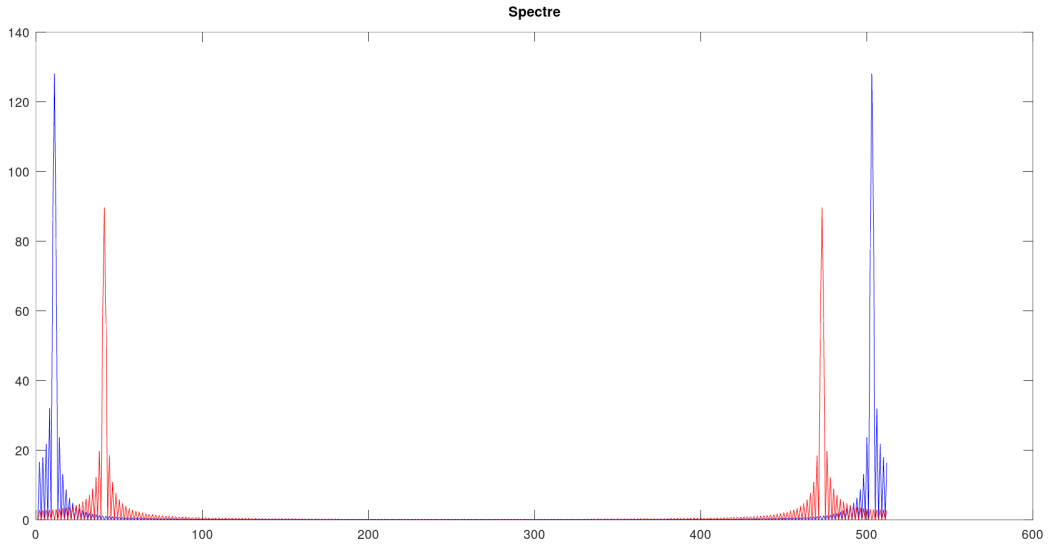

Рис. 6: График спектров синусоидальных сигналов.

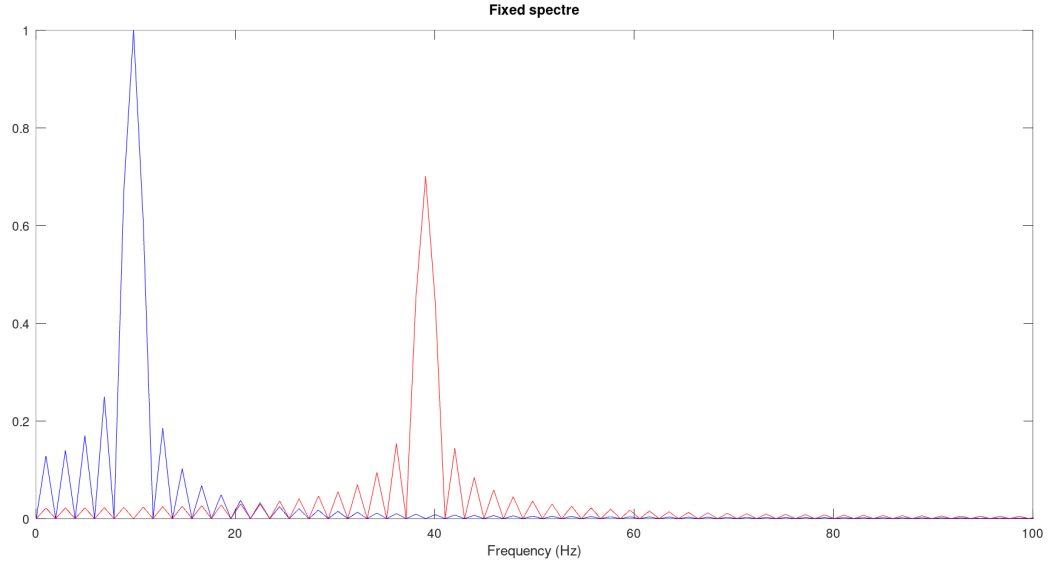

Рис. 7: Исправленный график спектров синусоидальных сигналов.

2.4 Амплитудная модуляция

1. Создали каталог `modulation` и в нем файл `am.m`. В файле реализовали программу, которая определяет спектр сигнала при амплитудной модуляции и строит соответствующие графики (Рисунки 8, 9):

```
% modulation/am.m
% Создание каталогов signal и spectre для размещения 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';
```

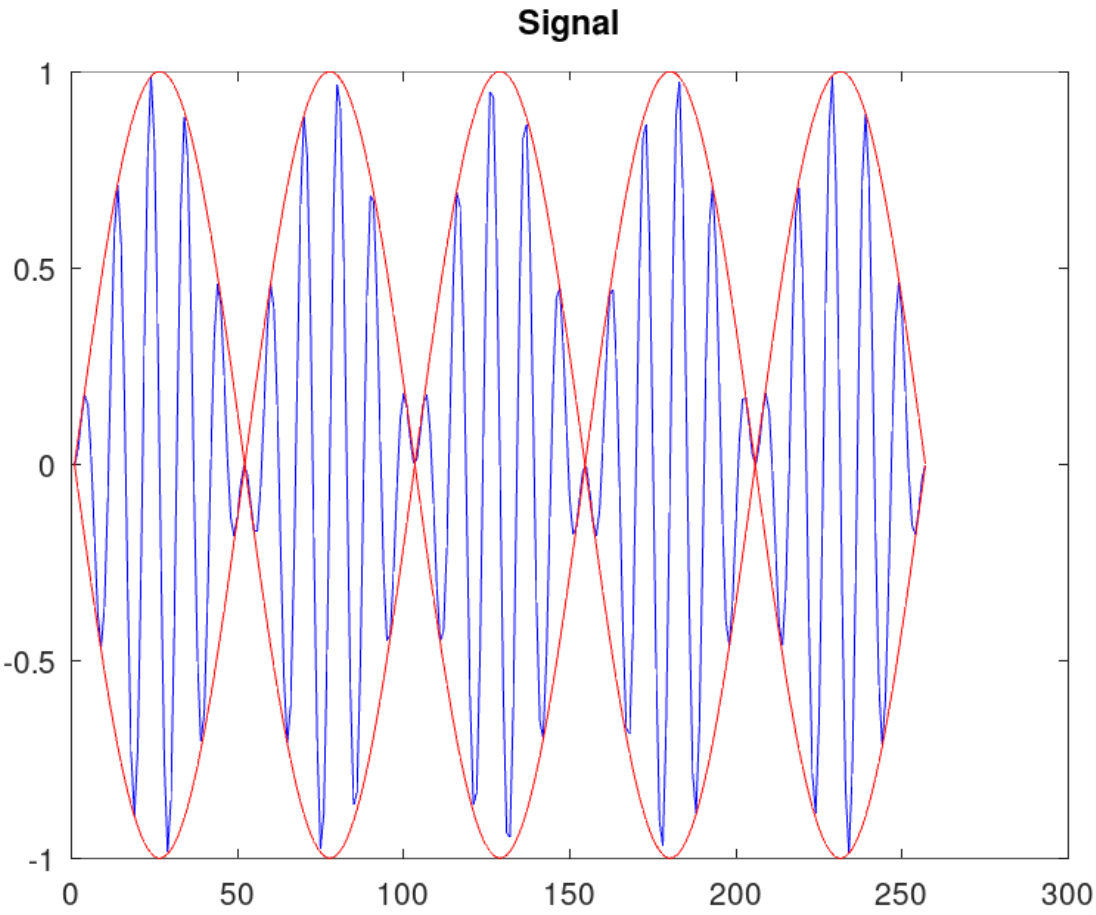

Рис. 8: Сигнал и огибающая при амплитудной модуляции.

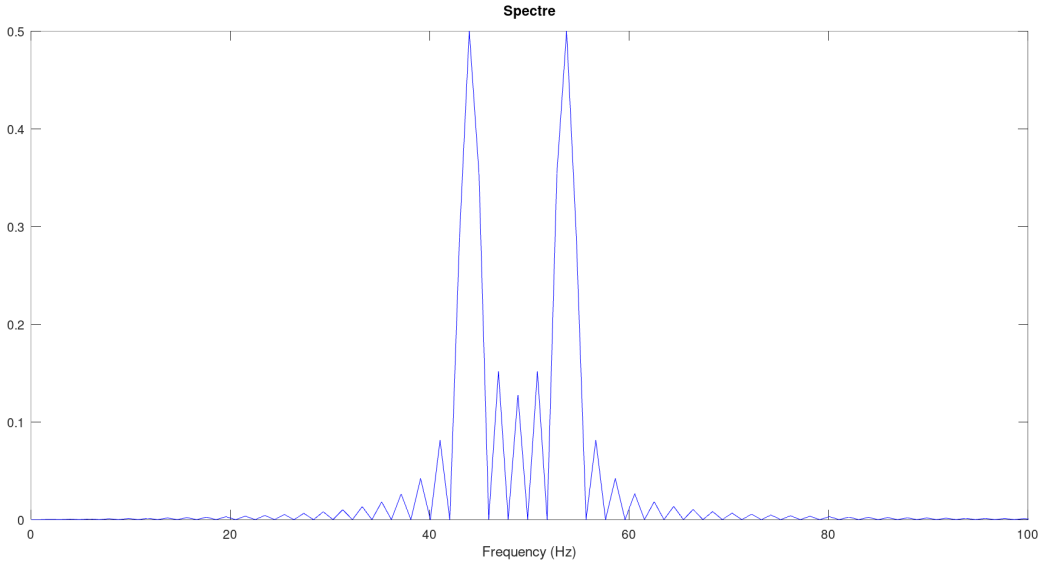

Рис. 9: Спектр сигнала при амплитудной модуляции.

2.5 Кодирование сигнала. Исследование свойства самосинхронизации сигнала

1. Создали каталог `coding` и в нем файлы: `main.m`, `maptowave.m`, `unipolar.m`, `ami.m`, `bipolarnrz.m`, `bipolarrz.m`, `manchester.m`, `diffmanc.m`, `calcspectre.m`.

2. Листинги каждого файла приведены ниже:

   - В файле `maptowave.m`:

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

   - В файле `unipolar.m`:

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

   - В файле `ami.m`:

     ```
     % 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);
     ```

   - В файле `bipolarnrz.m`:

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

   - В файле `bipolarrz.m`:

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

   - В файле `manchester.m`:

     ```
     % 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);
```

- В файле `diffmanc.m`:

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

- В файле `calcspectre.m`:

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

- В файле `main.m`:

```
% coding/main.m
% Подключение пакета signal:
pkg load signal;
% Входная кодовая последовательность:
data=[0 1 0 0 1 1 0 0 0 1 1 0];
% Входная кодовая последовательность для проверки свойства
↪  самосинхронизации:
data_sync=[0 0 0 0 0 0 0 1 1 1 1 1 1 1];
% Входная кодовая последовательность для построения спектра сигнала:
data_spectre=[0 1 0 1 0 1 0 1 0 1 0 1 0 1];
% Создание каталогов signal, sync и spectre для размещения графиков:
mkdir 'signal';
mkdir 'sync';
mkdir 'spectre';
axis("auto");

% 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';

% 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';

% 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';
```

3. После запуска главного скрипта `main.m` получены файлы с графиками кодированного сигнала: графики кодированного сигнала (Рис. 10-15), графики, иллюстрирующие свойства самосинхронизации (Рис. 16-21), графики спектров сигналов (Рис. 22-27):

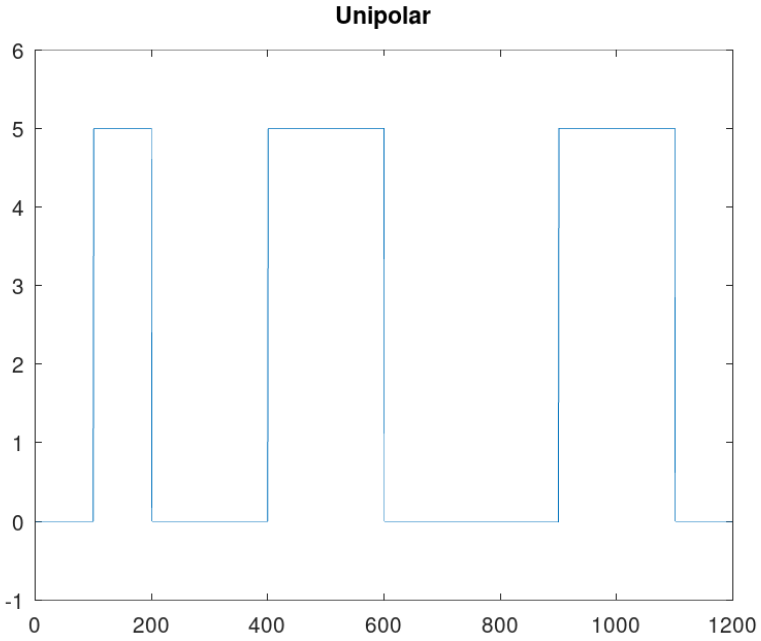

Рис. 10: Униполярное кодирование.

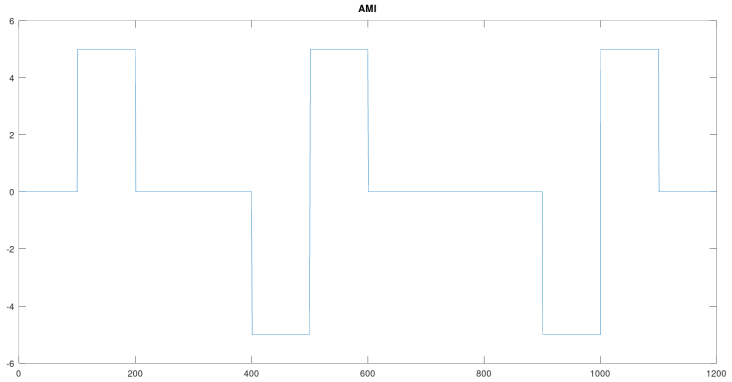

Рис. 11: Кодирование AMI.

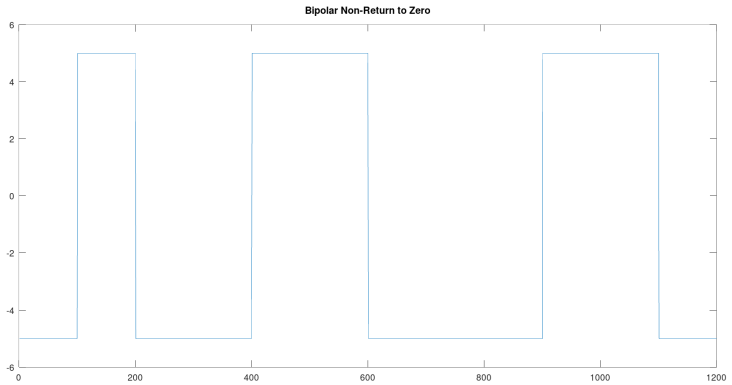

Рис. 12: Кодирование NRZ

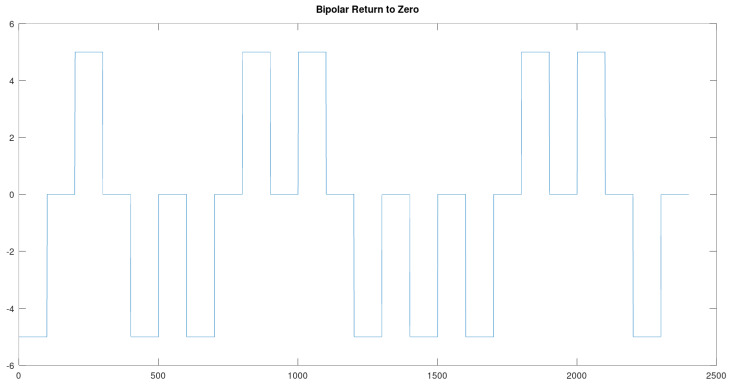

Рис. 13: Кодирование RZ

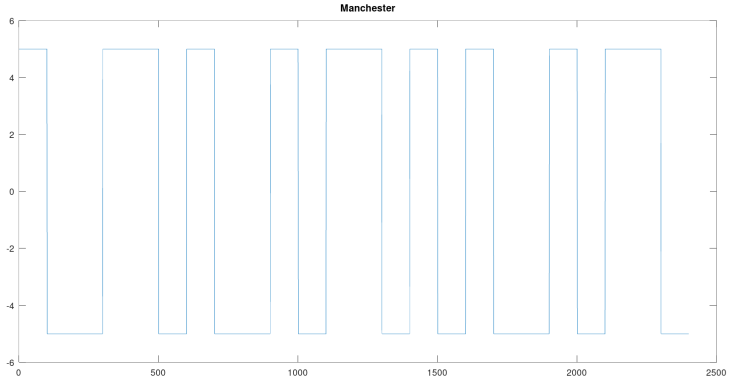

Рис. 14: Манчестрерское кодирование.

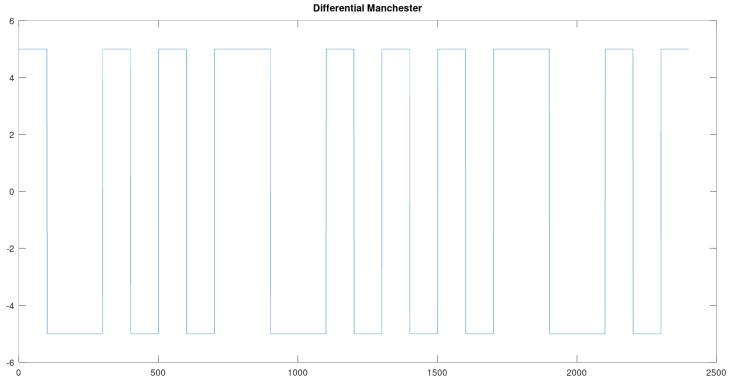

Рис. 15: Дифференциальное манчестерское кодирование.

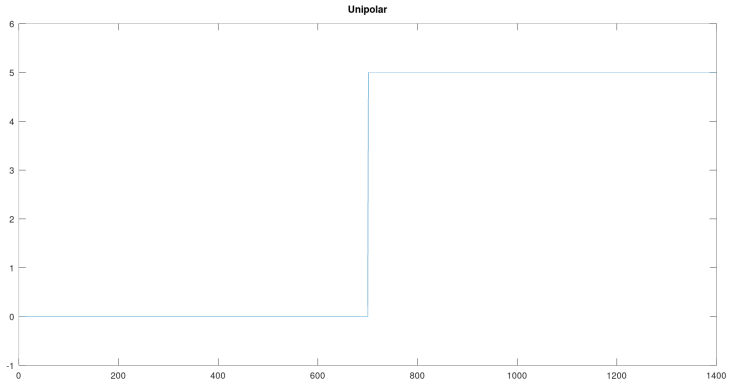

Рис. 16: Униполярное кодирование: нет синхронизации.

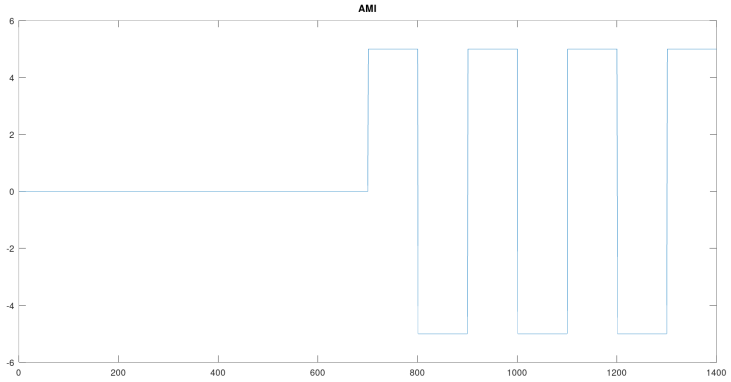

Рис. 17: Кодирование AMI: самосинхронизация при наличии сигнала.

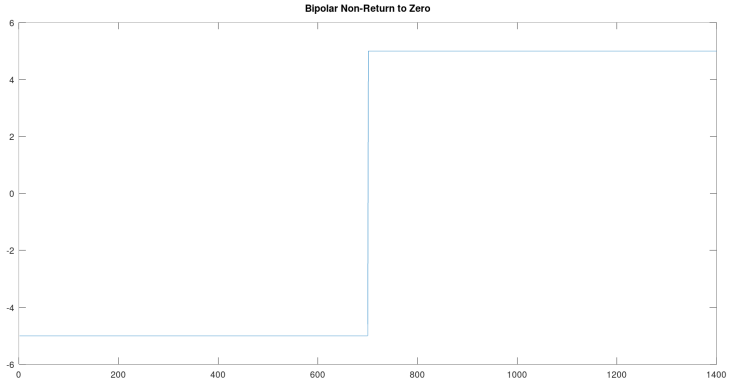

Рис. 18: Кодирование NRZ: нет самосинхронизации.

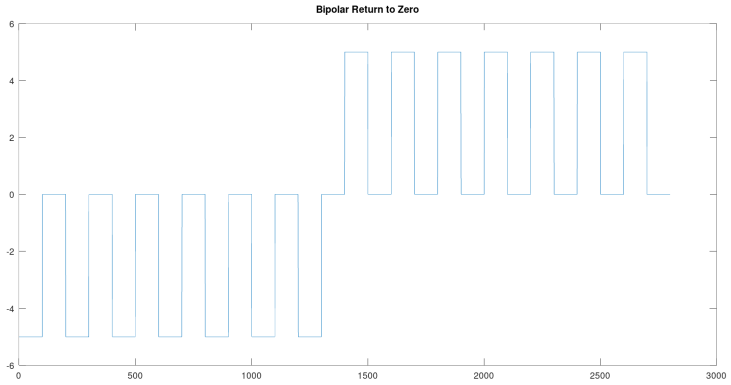

Рис. 19: Кодирование RZ: есть самосинхронизация

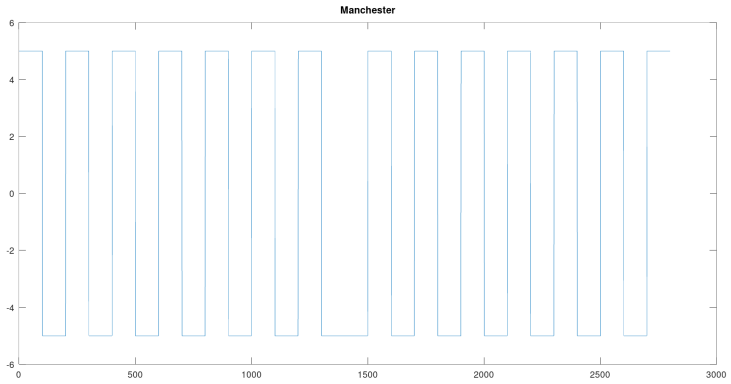

Рис. 20: Манчестрерское кодирование: есть самосинхронизация

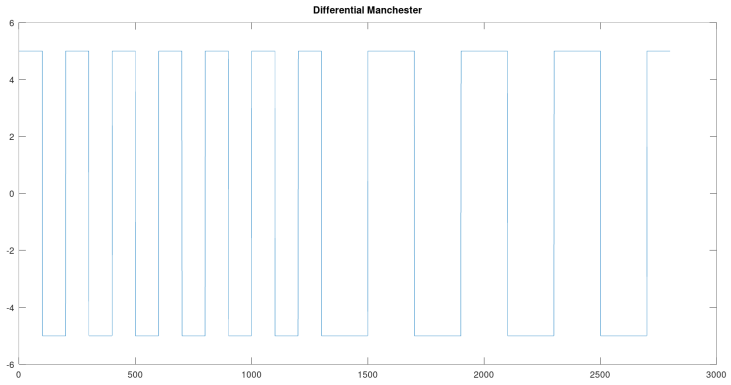

Рис. 21: Дифференциальное манчестерское кодирование: есть самосинхронизация

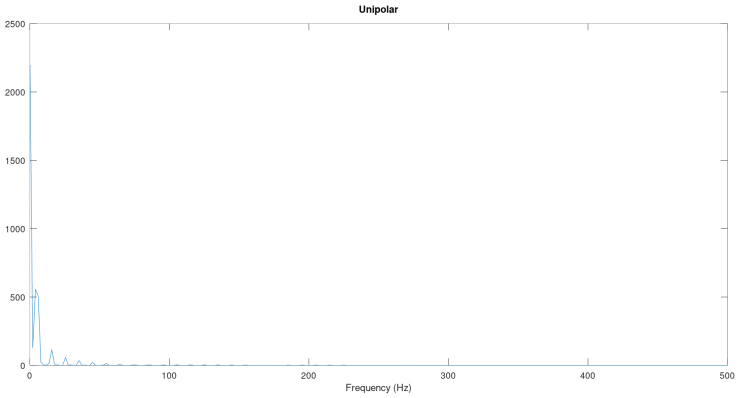

Рис. 22: Униполярное кодирование: спектр сигнала

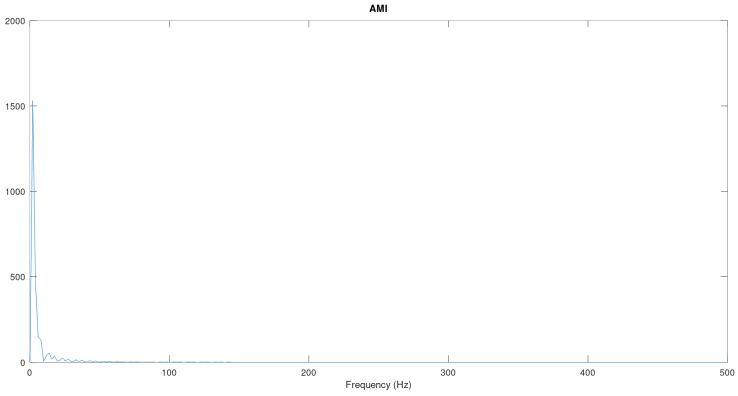

Рис. 23: Кодирование AMI: спектр сигнала

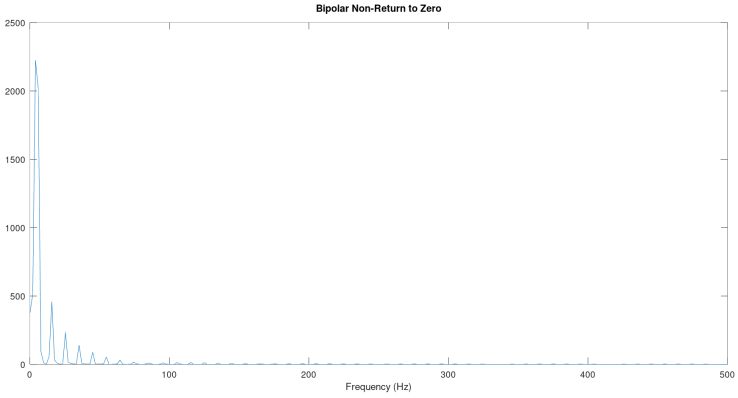

Рис. 24: Кодирование NRZ: спектр сигнала

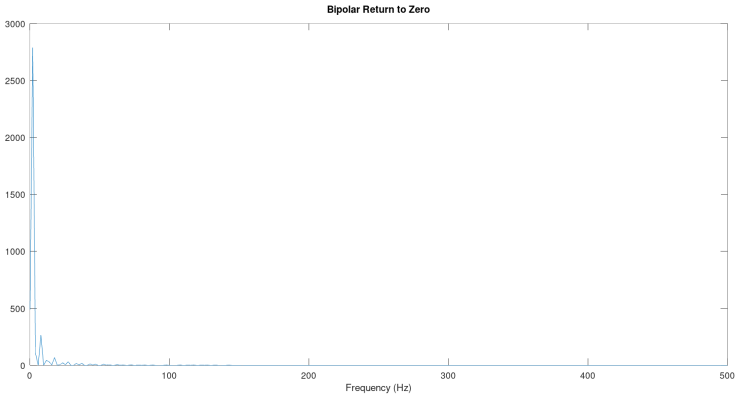

Рис. 25: Кодирование RZ: спектр сигнала

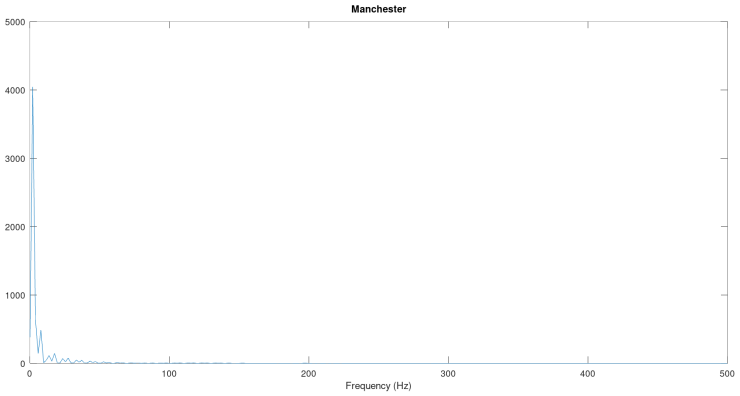

Рис. 26: Манчестрерское кодирование: спектр сигнала

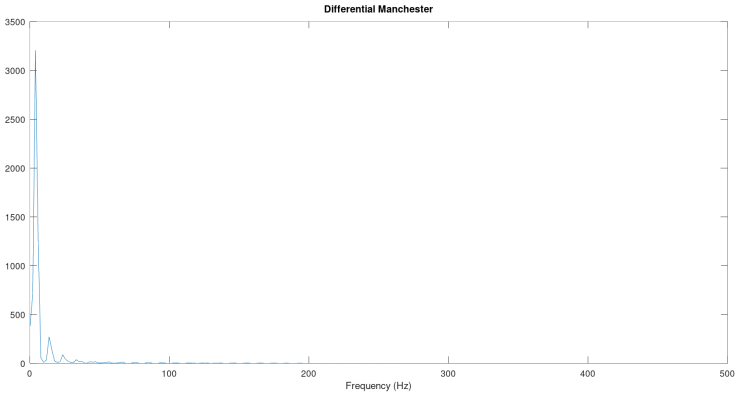

Рис. 27: Дифференциальное манчестерское кодирование: спектр сигнала

3 Выводы

В рамках выполнения лабораторной работы изучили методы кодирования и модуляции сигналов с помощью высокоуровнего языка программирования Octave, в том числе познакомились с принципами модуляции сигнала на примере аналоговй амплитудной модуляции, исседовали свойства самосинхронизации сигнала.