Отчёт по лабораторной работе 1

Методы кодирования и модуляция сигналов

Цвелев С.А. НПИбд-02-22

Содержание

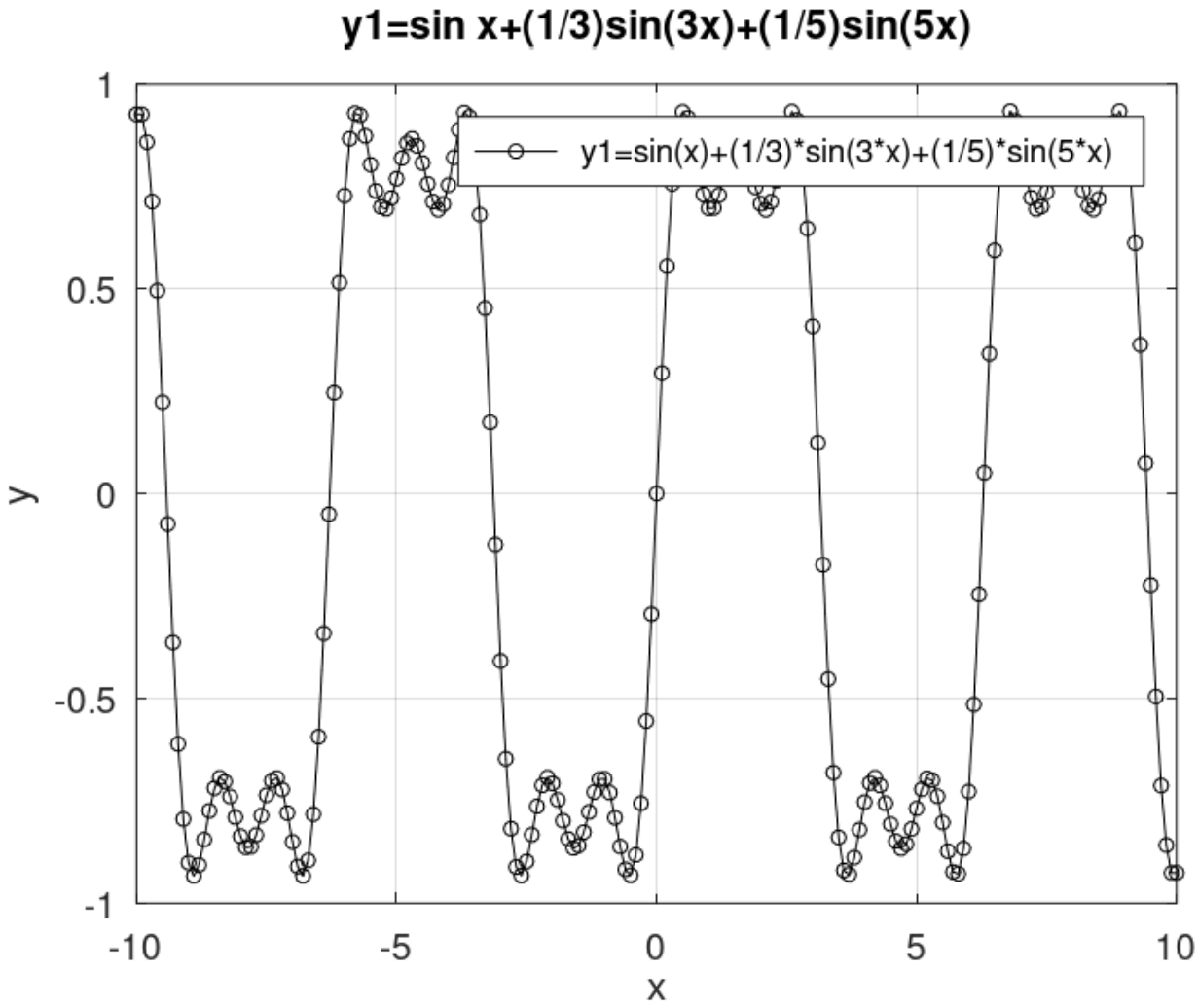
1 Цель работы

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.

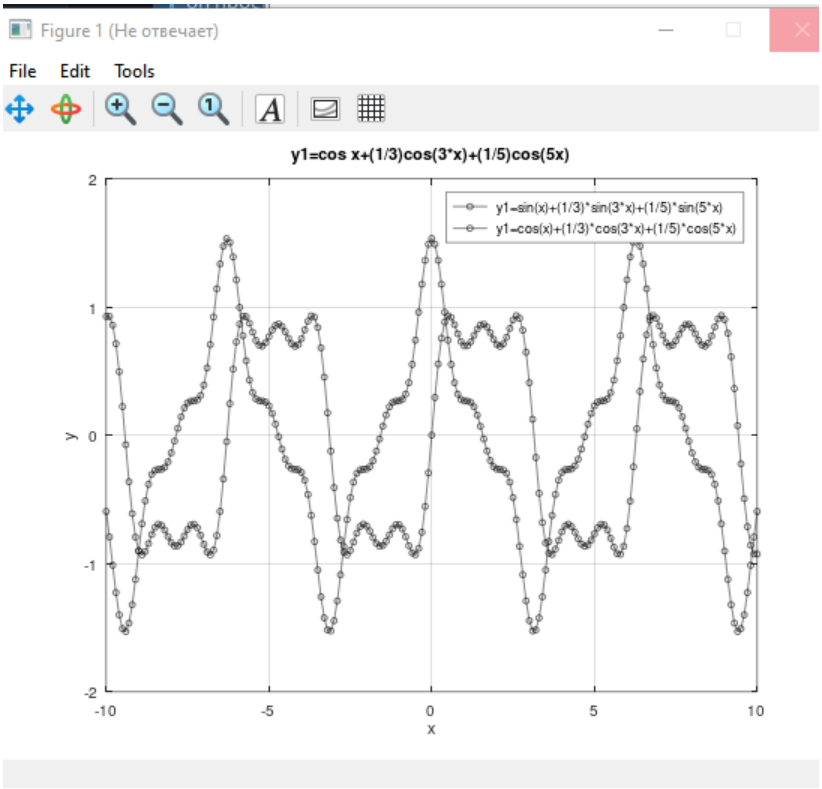
2 Ход работы

Первым делом, мы строим график функции, а затем выводим получившееся изображение в файлы формата eps и png.

```
% Формирование массива x:
x=-10:0.1:10;
% Формирование массива y.
yl=sin(x)+1/3*sin(3*x)+1/5*sin(5*x);
% Построение графика функции:
plot(x,yl, "-ok; yl=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('yl=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");
```

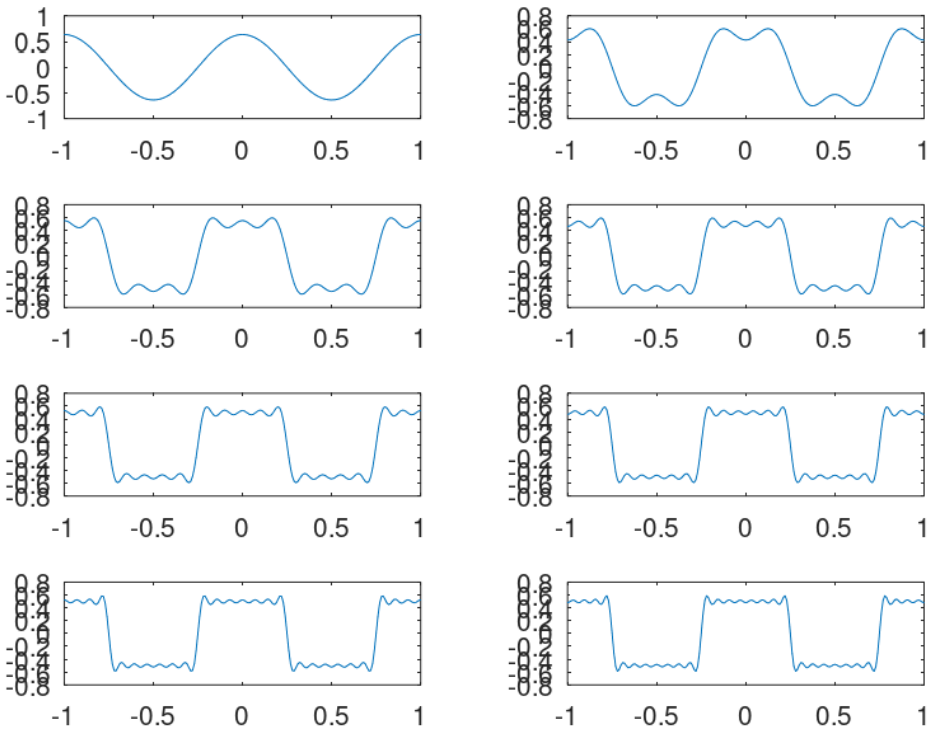


Далее, мы строим тут же второй график, сделав функцию через косинусы. Изображение получилось низкого качества из-за того, что программа вылетала при запуске.



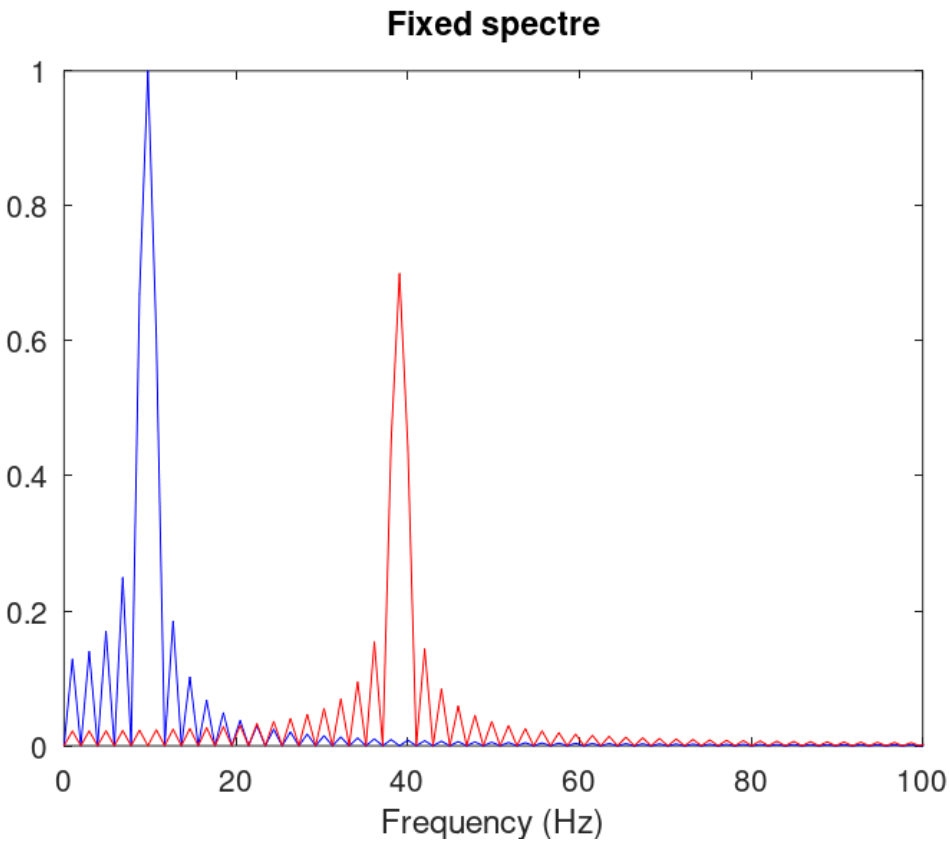
Далее, мы создали новый сценарий, назвав meandr.m. Как и в прошлый раз, график выводился в отдельный файл.

```
% meandr.m
% количество отсчетов (гармон
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 ря
Am=2/pi ./ nh;
Am(2:2:end) = -Am(2:2:end);
% массив гармоник
harmonics=cos(2 * pi * nh' *
% массив элементов ряда:
sl=harmonics.*repmat(Am',1,le
% Суммирование ряда:
s2=cumsum(sl);
% Построение графиков
for k=1:N
  subplot(4,2,k)
  plot(t, s2(k,:))
end
print("meandr.png");
```

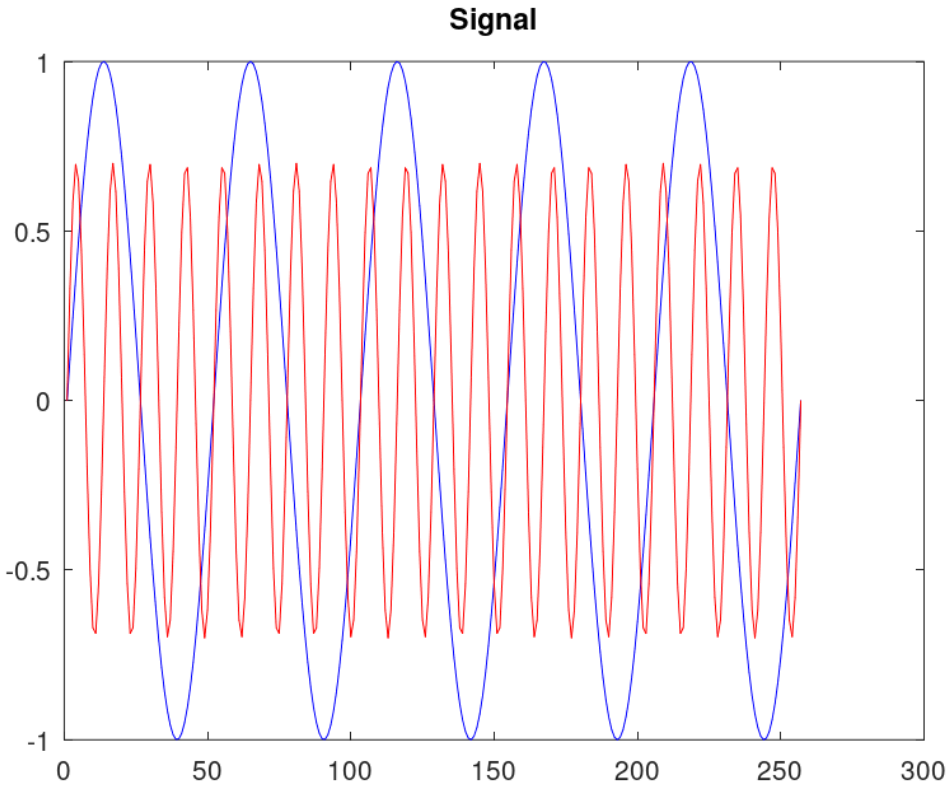


Мы создали отдельный каталог со сценарием spectre.m. Создались отдельные подкаталоги, куда выводились сигналы.

```
% spectrel/spectre.m

%Создание каталогов signal и spectre для размещения г
mkdir 'signal';
mkdir 'spectre';

% Длина сигнала (с):
tmax = 0.5;
% Частота дискретизации (Гц) (количество отсчетов):
fd = 512;
% Частота первого сигнала (Гц):
f1 = 10;
% Частота второго сигнала (Гц):
f2 = 40;
% Амплитуда первого сигнала:
a1 = 1;
% Амплитуда второго сигнала:
a2 = 0.7;
% Массив отсчетов времени:
t = 0:1./fd:tmax;
% Спектр сигнала:
fd2 = fd/2;

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

% График 1-го сигнала:
plot(signal1, 'b');
% График 2-го сигнала:
hold on
plot(signal2,'r');
hold off
title('Signal');

% Экспорт графика в файл в каталоге signal:
print 'signal/spectre.png';

% 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';
```

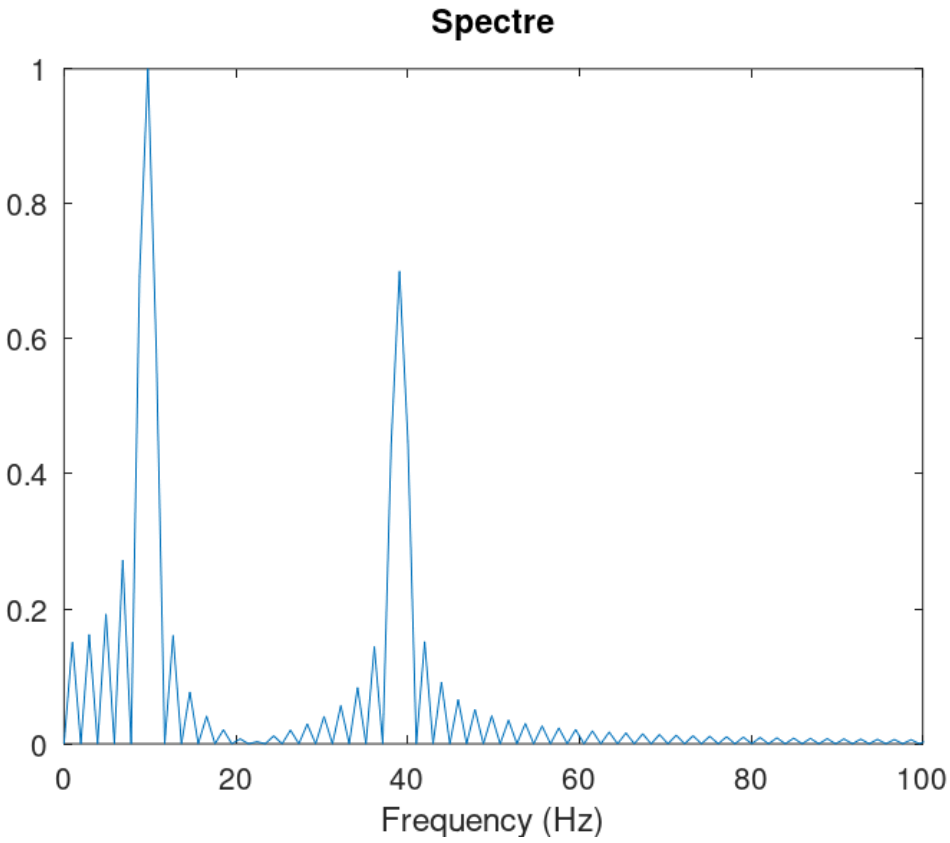


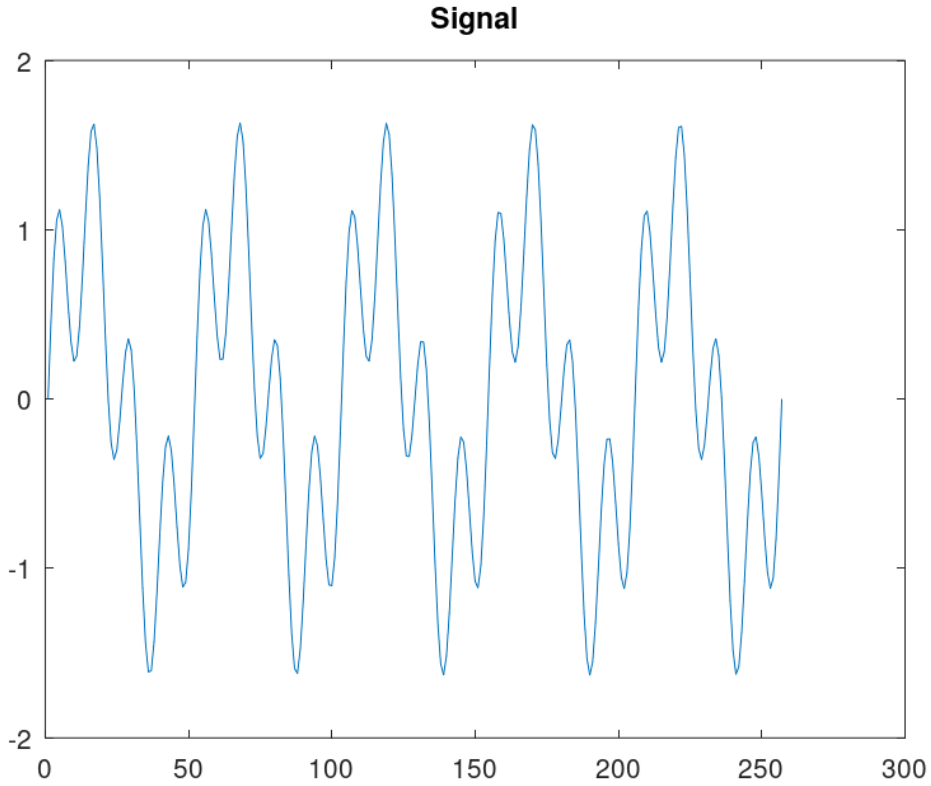
Код был изменён для обновленного вывода, изображение выше.



В файле spectre_sum.m мы подсчитывали спектр суммы сигналов.

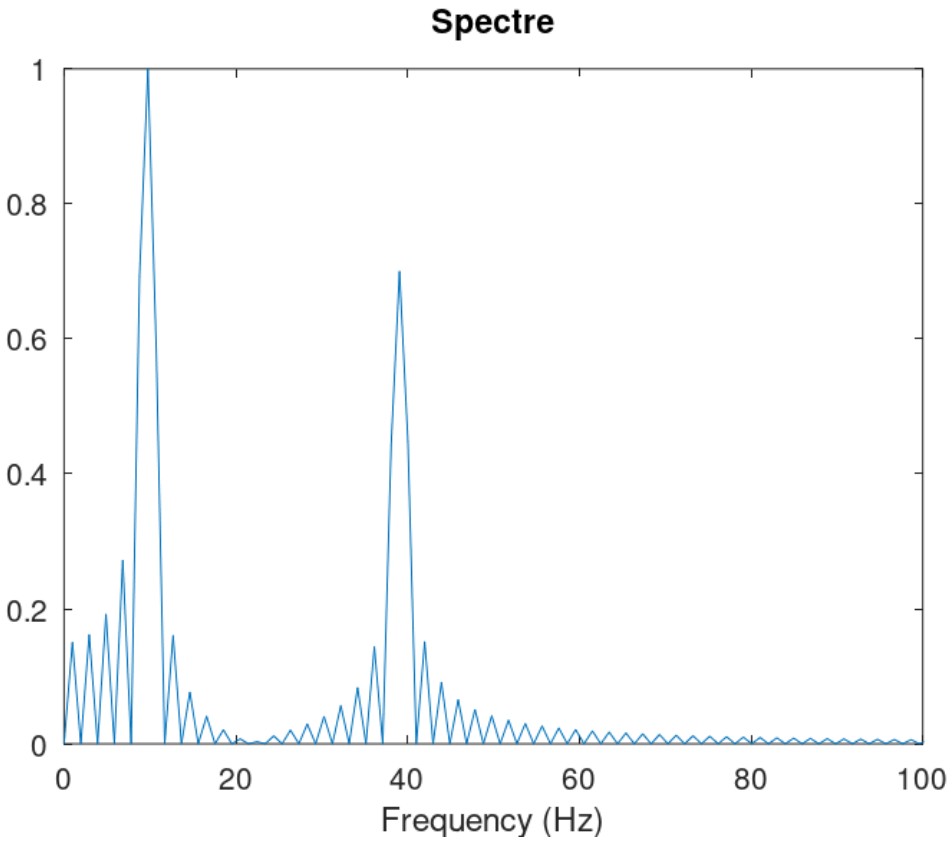
```
% spectre_sum/spectre_sum.m
% Создание каталогов signal и spectre для размещения графиков:
mkdir 'signal';
mkdir 'spectre';
% Длина сигнала (с):
tmax = 0.5;
% Частота дискретизации (Гц) (количество отсчётов):
fd = 512;
% Частота первого сигнала (Гц):
f1 = 10;
% Частота второго сигнала (Гц):
f2 = 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';
```

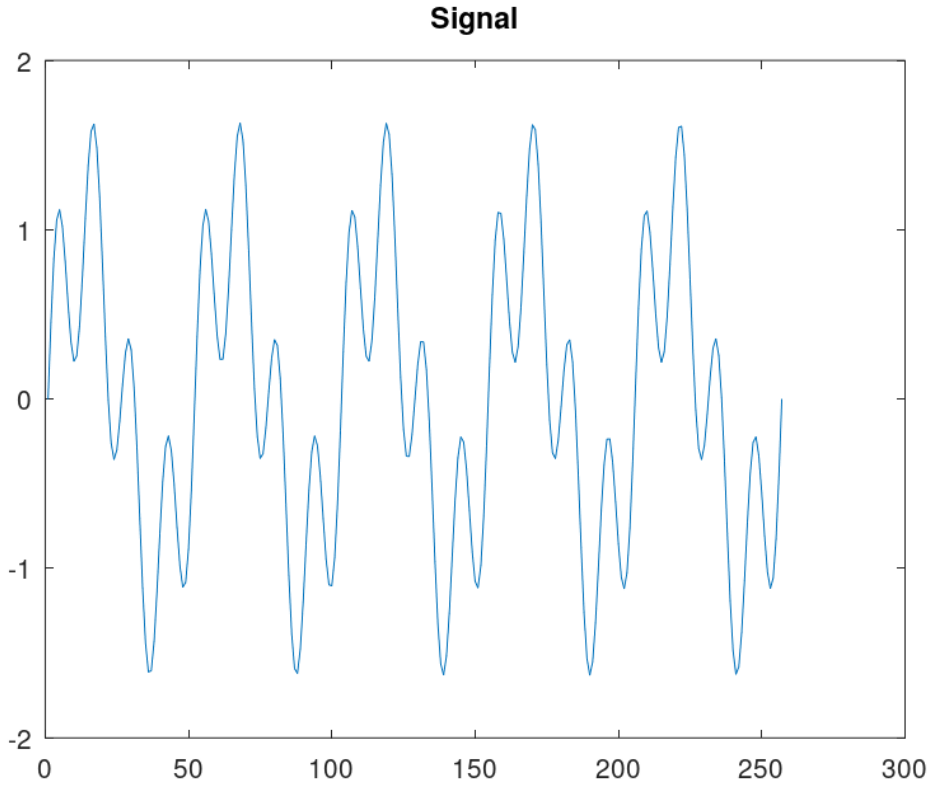




Следующим заданием мы выводили спектр произведения, в файле am (в каталоге modulation)

```
% 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';
```





Последним заданием было кодирование сигнала. Для начала мы проверили наличие пакета signal через интерпретатор. Убедившись, что он установлен, мы создали сразу несколько файлов в каталоге coding - main.m,

maptowave.m,unipolar.m,ami.m,bipolarnrz.m,bipolarrz.m,manchester.m,

diffmanc.m, calcspectre.m

В главном сценарии мы задали вызов функций для построения графиков модуляций кодированных сигналов, а также для построения спектров (к сожалению, файлы спектров у меня почему-то не создались).

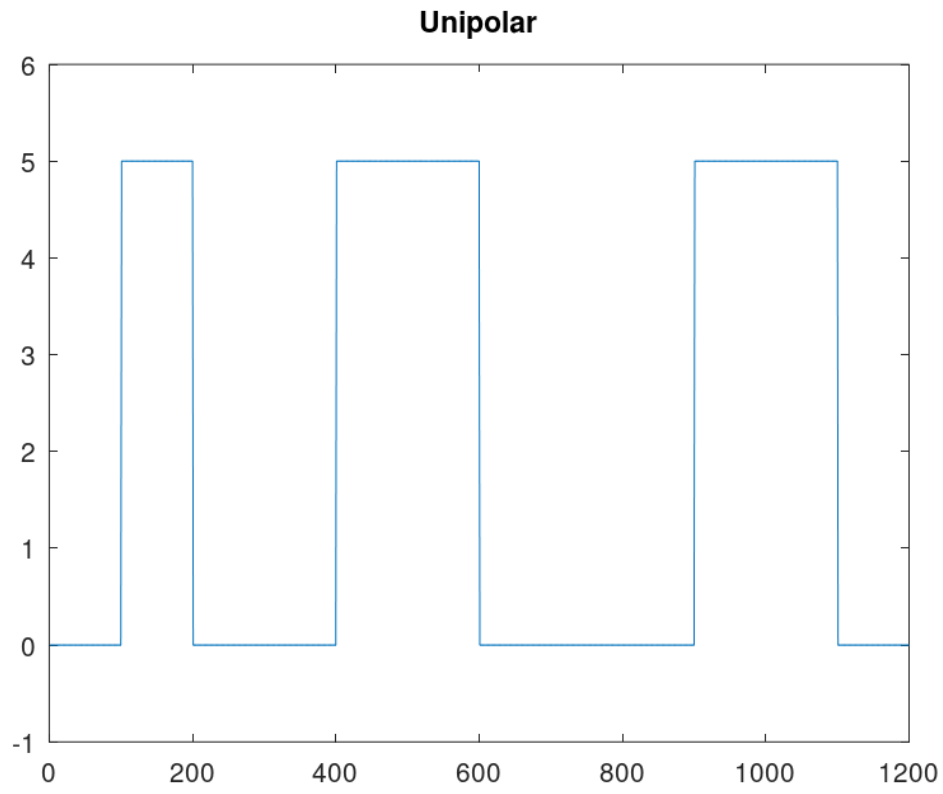
```
% coding/main.m
% Подключение пакета signal:
pkg load signal;

% 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");

% data
% 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';

% data_sync
% Униполярное кодирование
wave=unipolar(data_sync);
plot(wave);
```

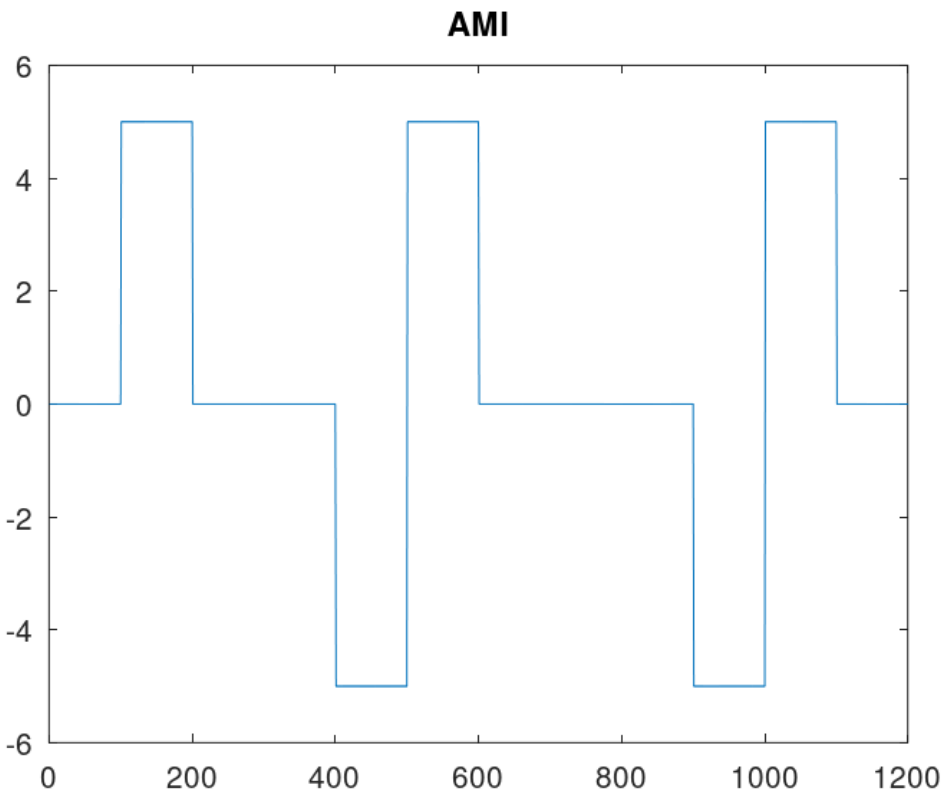
(Дальше были вызовы подобных функций для всех остальных файлов)

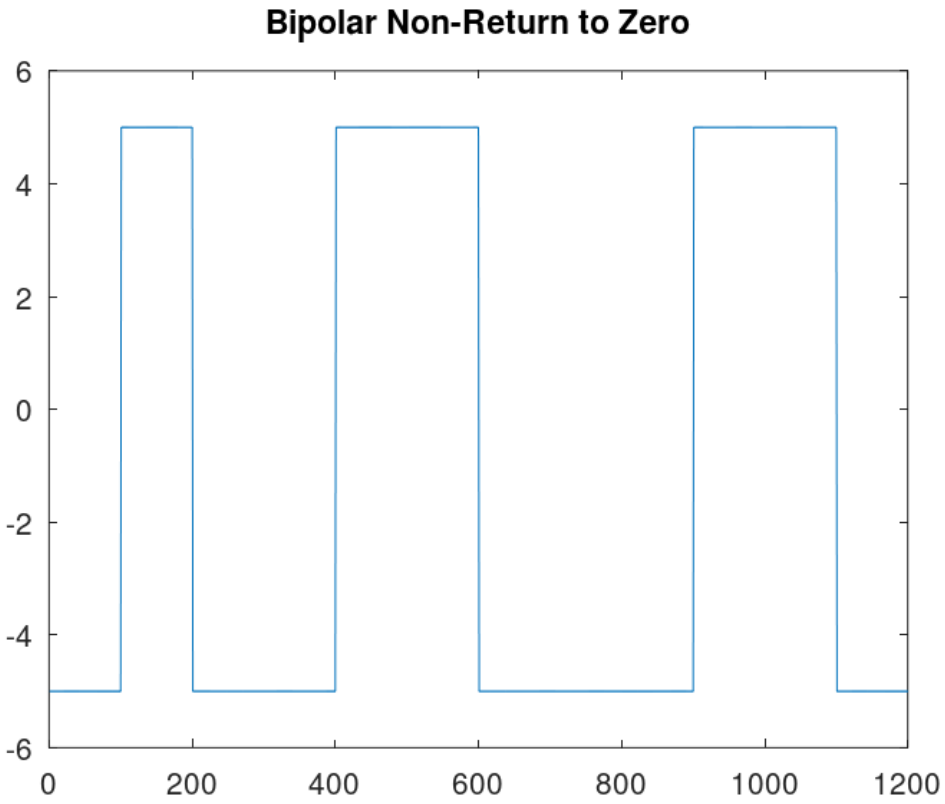
В самих файлах были подобные коды:

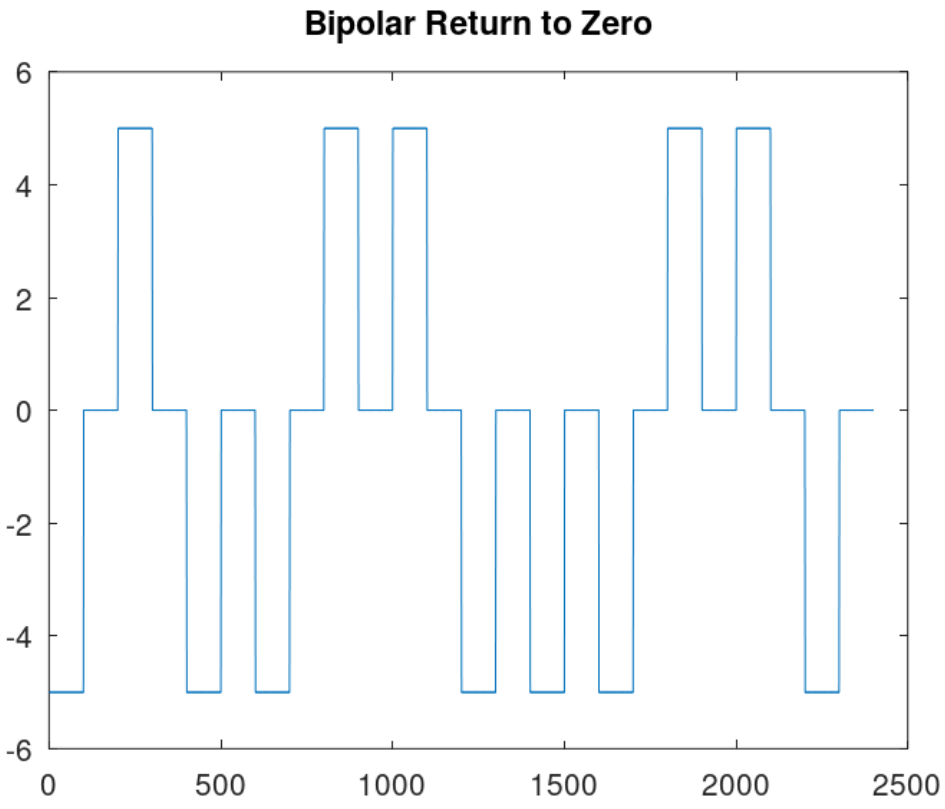
```
% coding/maptowave.m
function wave=maptowave(data)
  data=upsample(data,100);
  wave=filter(5*ones(1,100),1,data);
```

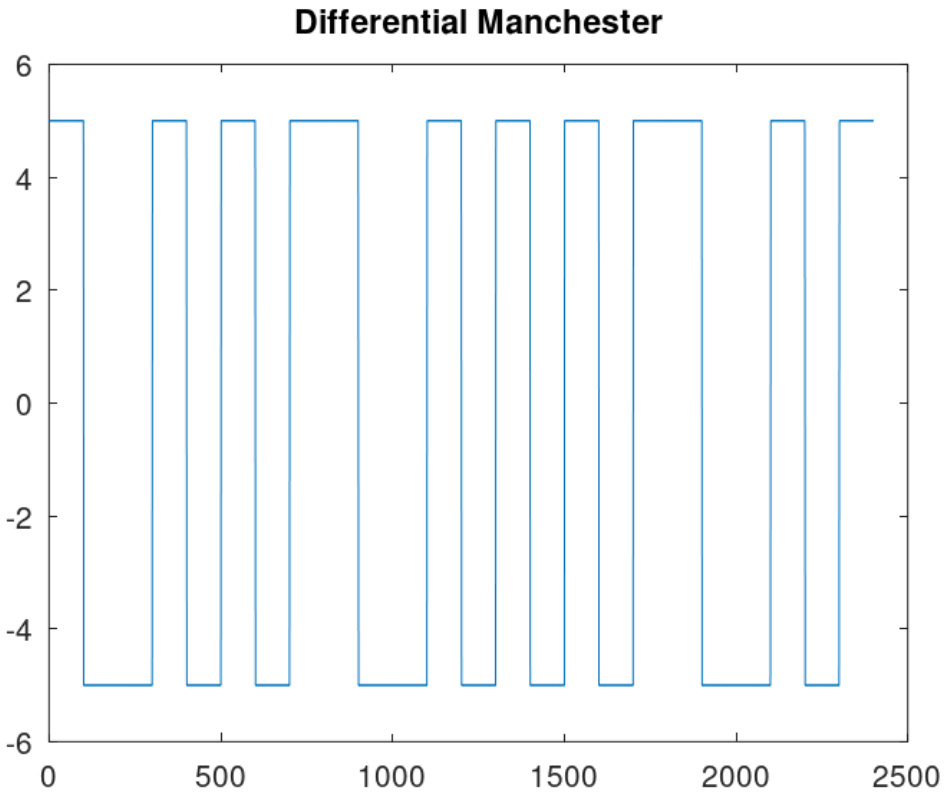
Итог:


Unipolar
6
5
4
3
2
1
0
-1
0
200
400
600
800
1000
1200

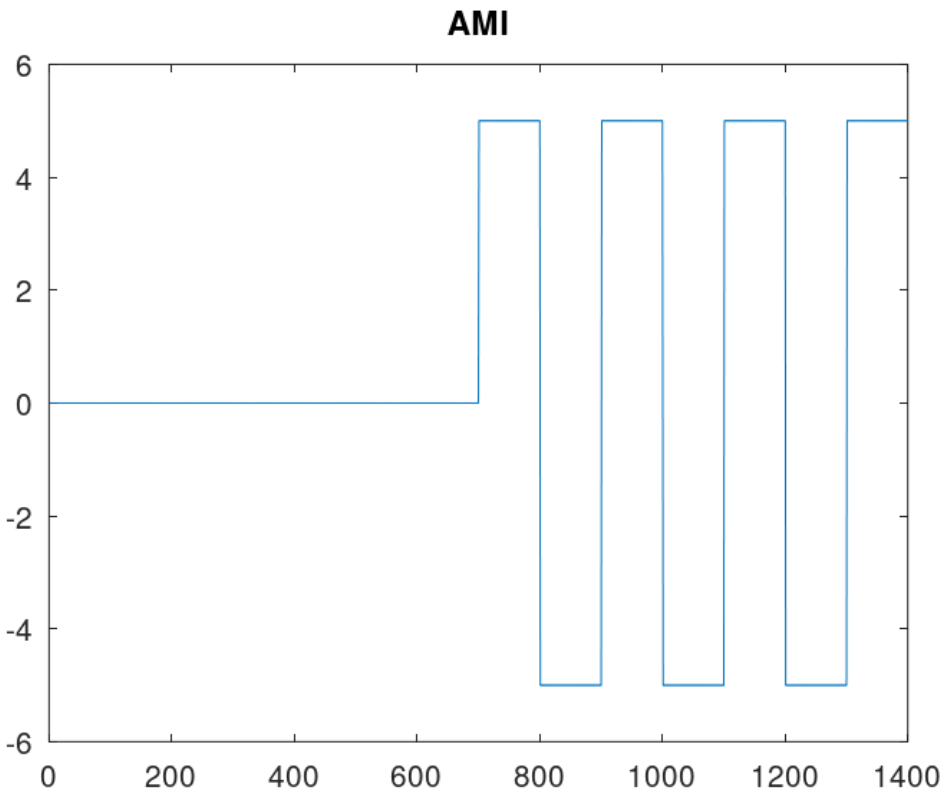

AMI
6
4
2
0
-2
-4
-6
0
200
400
600
800
1000
1200

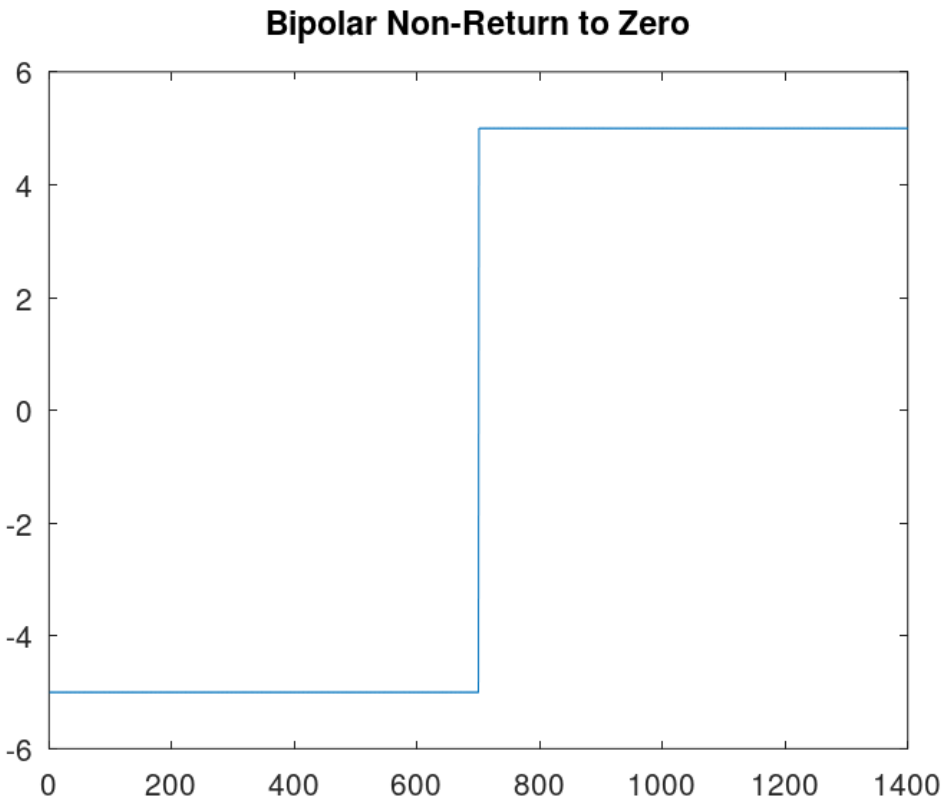

Bipolar Non-Return to Zero
6
4
2
0
-2
-4
-6
0
200
400
600
800
1000
1200

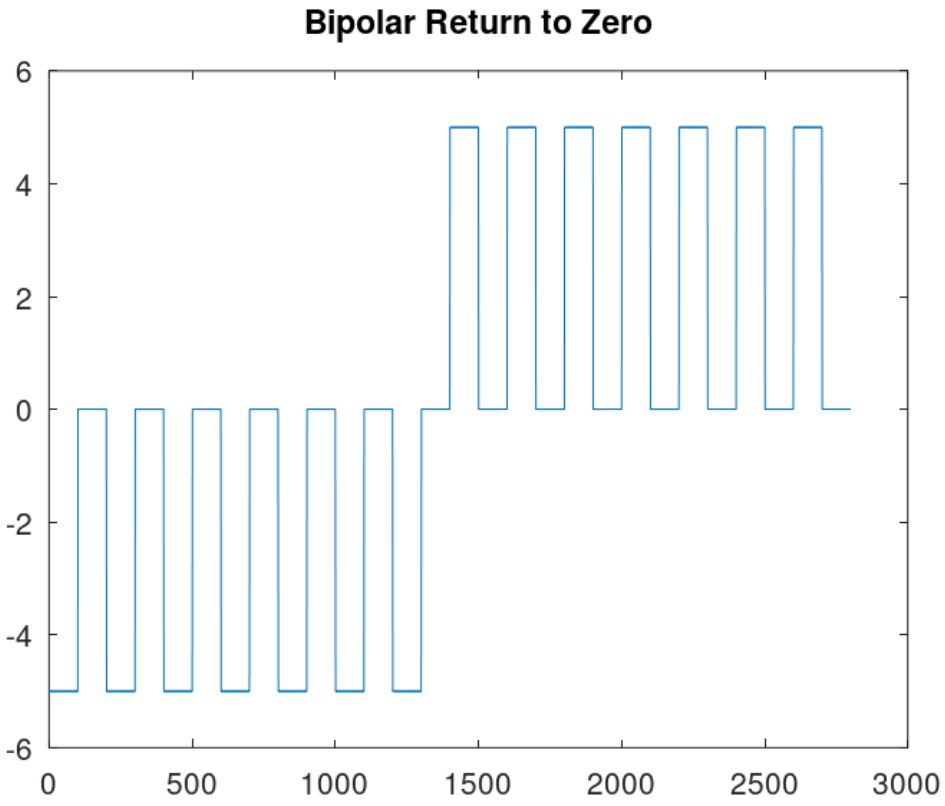

Bipolar Return to Zero
6
4
2
0
-2
-4
-6
0
500
1000
1500
2000
2500

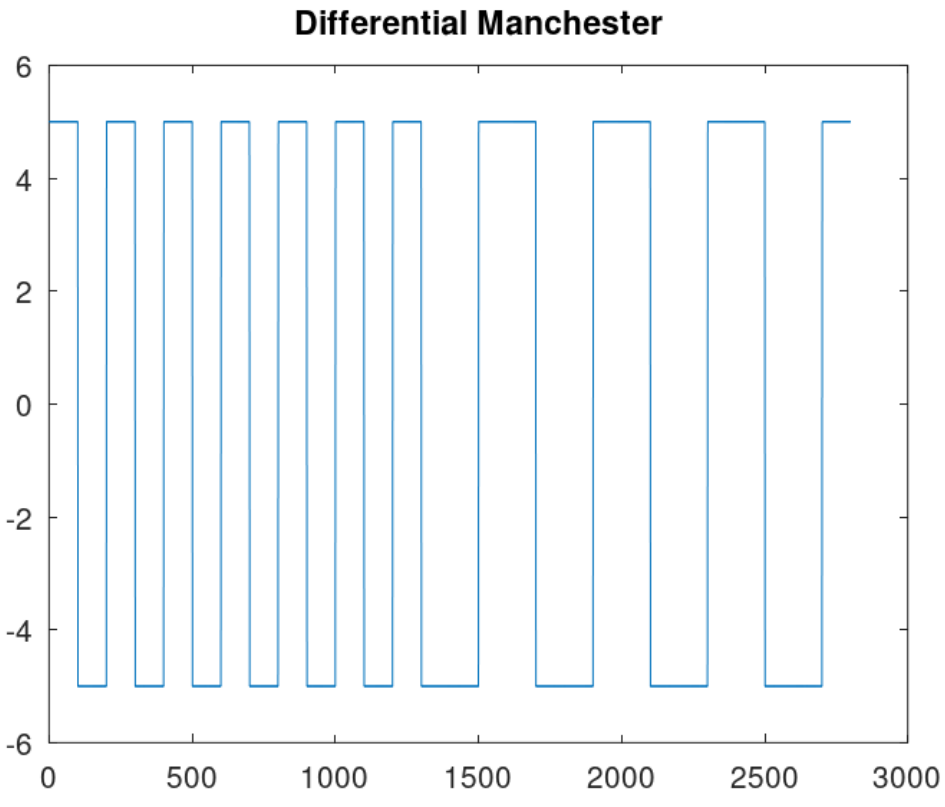


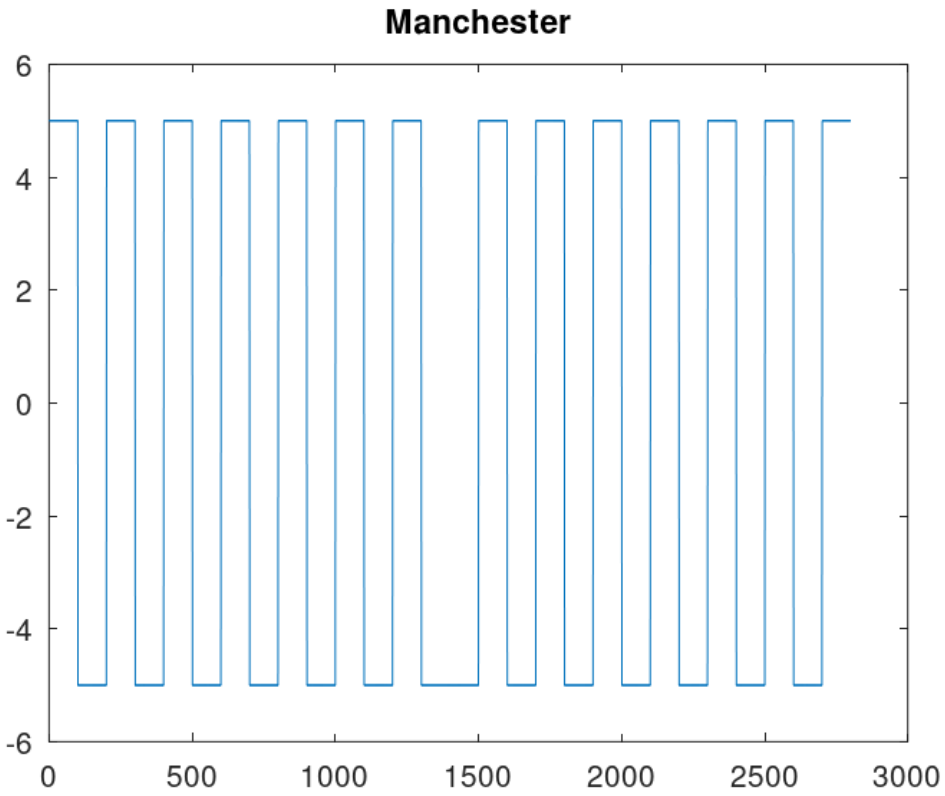
Собственно, сигналы.

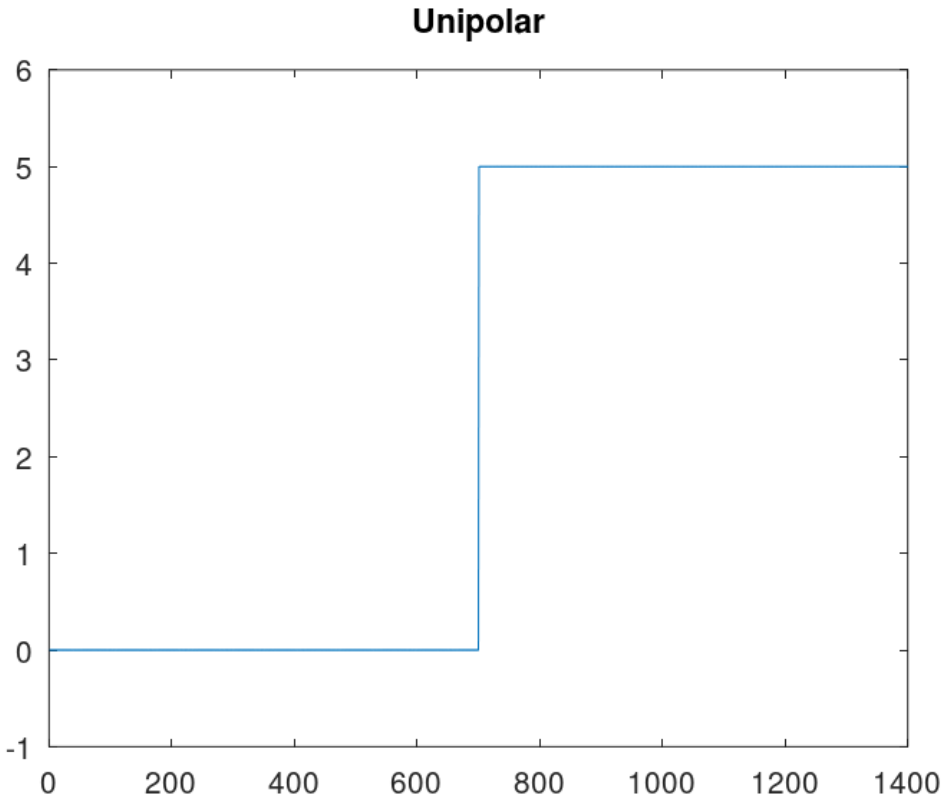

AMI
6
4
2
0
-2
-4
-6
0
200
400
600
800
1000
1200
1400


Bipolar Non-Return to Zero
6
4
2
0
-2
-4
-6
0
200
400
600
800
1000
1200
1400


Bipolar Return to Zero
6
4
2
0
-2
-4
-6
0
500
1000
1500
2000
2500
3000


Differential Manchester
6
4
2
0
-2
-4
-6
0
500
1000
1500
2000
2500
3000





Синхронизация.

3 Вывод

Мы изучили методы кодирования и модуляцию сигналов с помощью языка кодирования Octave, узнали, что такое спектр и параметры сигнала, а также продемонстрировали принципы модуляции сигнала.