

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ  
Факультет физико-математических и естественных наук  
Кафедра прикладной информатики и теории  
вероятностей

## **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1**

дисциплина: Сетевые Технологии

Студент: Оулед Салем

Яссин

Группа:НПИбд-02-20

МОСКВА

2022 г

## Цели работы

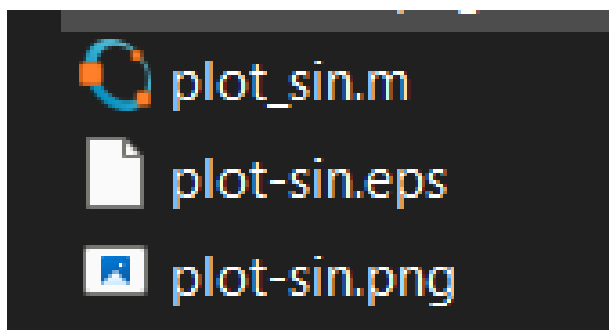
Изучение методов кодирования и модуляции сигналов с помощью высокоуровневого языка программирования Octave. Определение спектра и параметров сигнала. Демонстрация принципов модуляции сигнала на примере аналоговой амплитудной модуляции. Исследование свойства самосинхронизации сигнала.

### 1.3.1.1. Постановка задачи

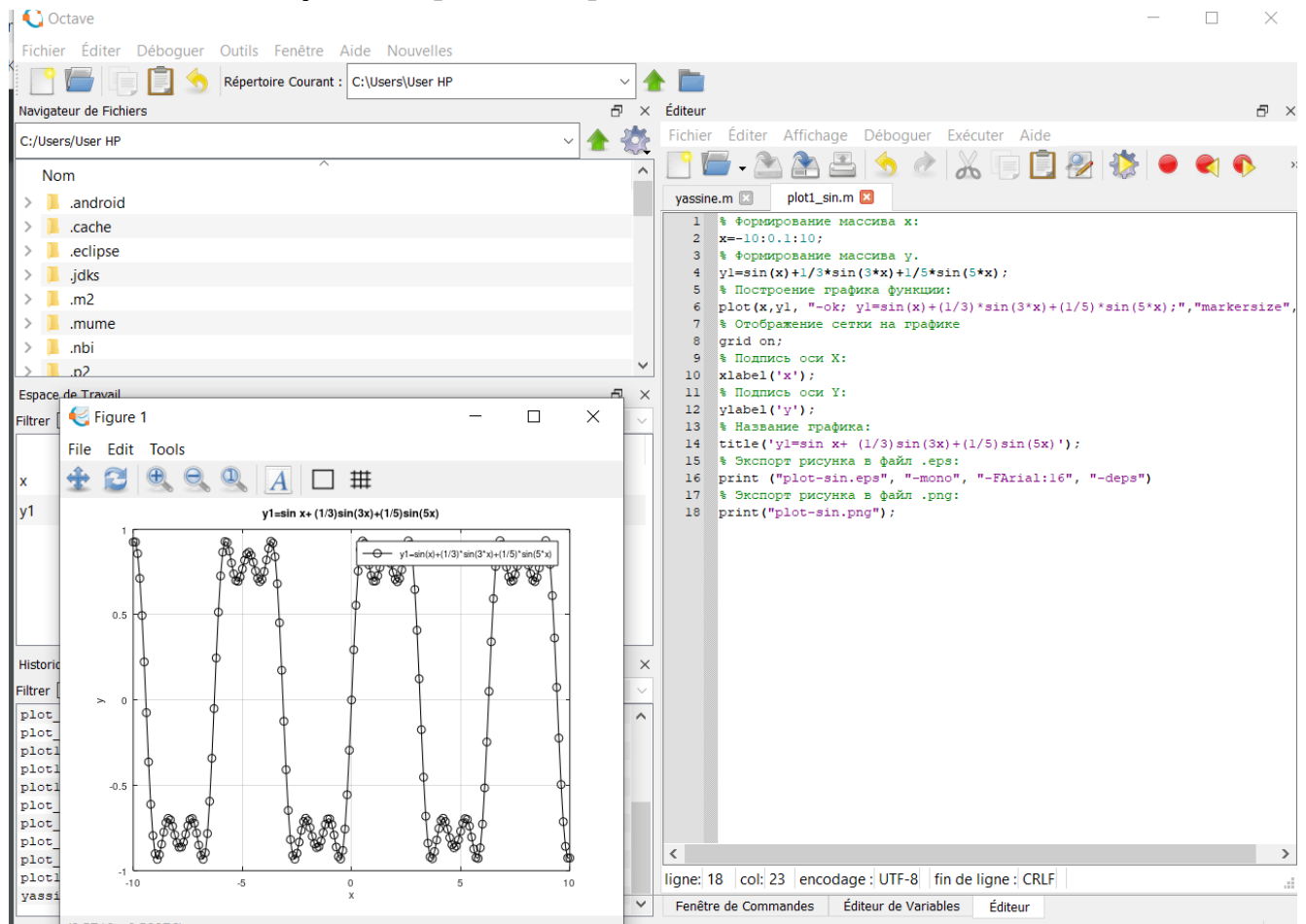
1. Построить график функции  $y = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$  на интервале  $[-10; 10]$ , используя Octave и функцию plot. График экспортировать в файлы формата .eps, .png.
2. Добавить график функции  $y = \cos x + 1/3 \cos 3x + 1/5 \cos 5x$  на интервале  $[-10; 10]$ . График экспортировать в файлы формата .eps, .png.

### 1.3.1.2. Порядок выполнения работы

1. Запустите в вашей ОС Octave с оконным интерфейсом.
2. Перейдите в окно редактора. Воспользовавшись меню или комбинацией клавиш `ctrl + n` создайте новый сценарий. Сохраните его в ваш рабочий каталог с именем, например, `plot_sin.m`.



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



**4. Запустите сценарий на выполнение (воспользуйтесь соответствующим меню окна редактора или клавишей F5 ). В качестве результата выполнения кода должно открыться окно с построенным графиком (рис. 1.1) и в вашем рабочем каталоге должны появиться файлы с графиками в форматах .eps, .png.**

Рис. 1.1. График функции  $y = \sin x + \frac{1}{3} \sin 3x + \frac{1}{5} \sin 5x$  на интервале  $[-10; 10]$

**5. Сохраните сценарий под другим названием и измените его так, чтобы на одном графике располагались отличающиеся по типу линий графики функций**

$y_1 = \sin x + \frac{1}{3} \sin 3x + \frac{1}{5} \sin 5x$ ,  $y_2 = \cos x + \frac{1}{3} \cos 3x + \frac{1}{5} \cos 5x$ ,  
например как изображено на рис. 1.2.

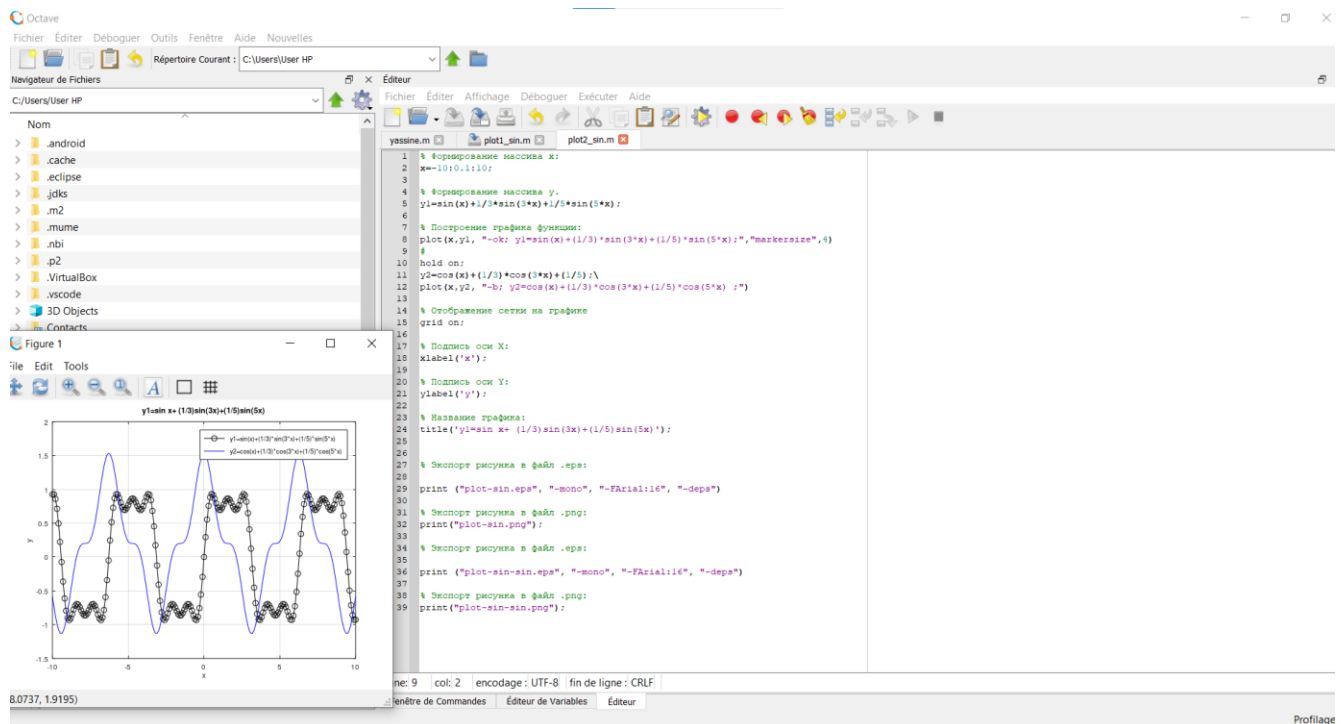


Рис. 1.2. График функций  $y_1$  и  $y_2$  на интервале  $[-10; 10]$

## 1.3.2. Разложение импульсного сигнала в частичный ряд Фурь

### 1.3.2.1. Постановка задачи

1. Разработать код m-файла, результатом выполнения которого являются графики меандра (рис. 1.3), реализованные с различным количе

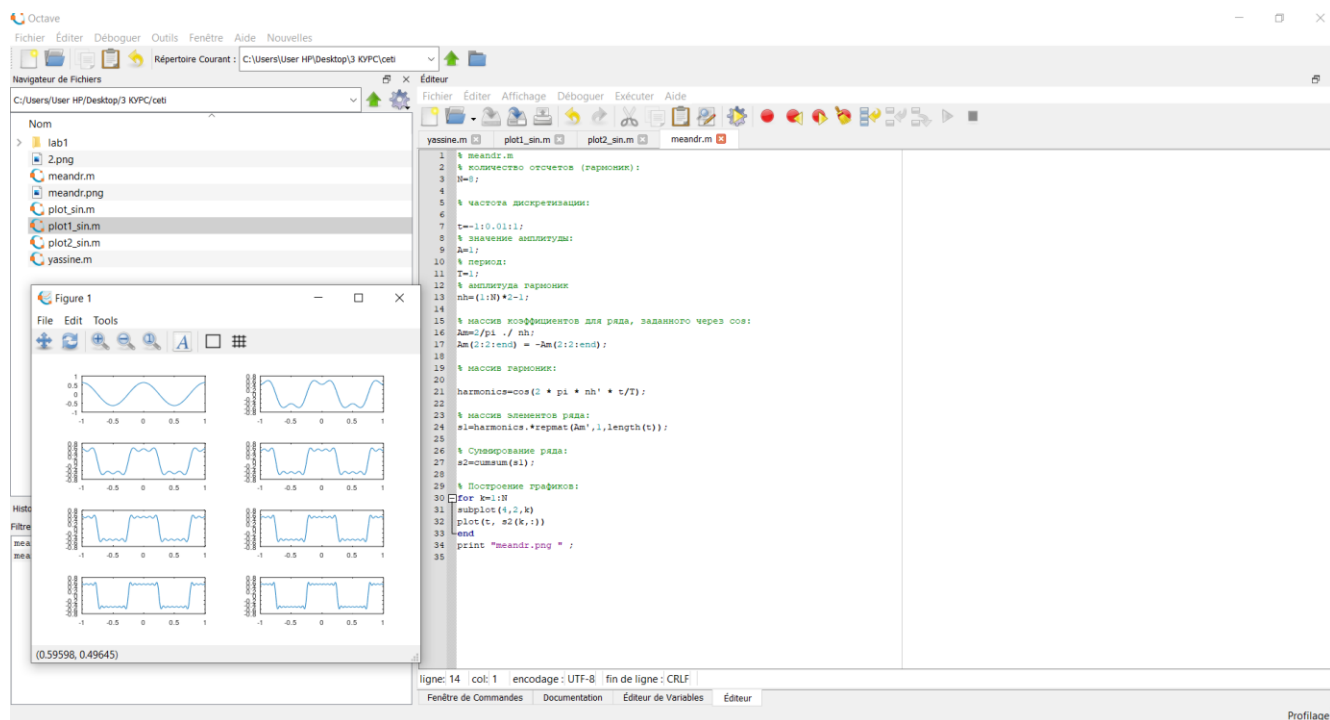
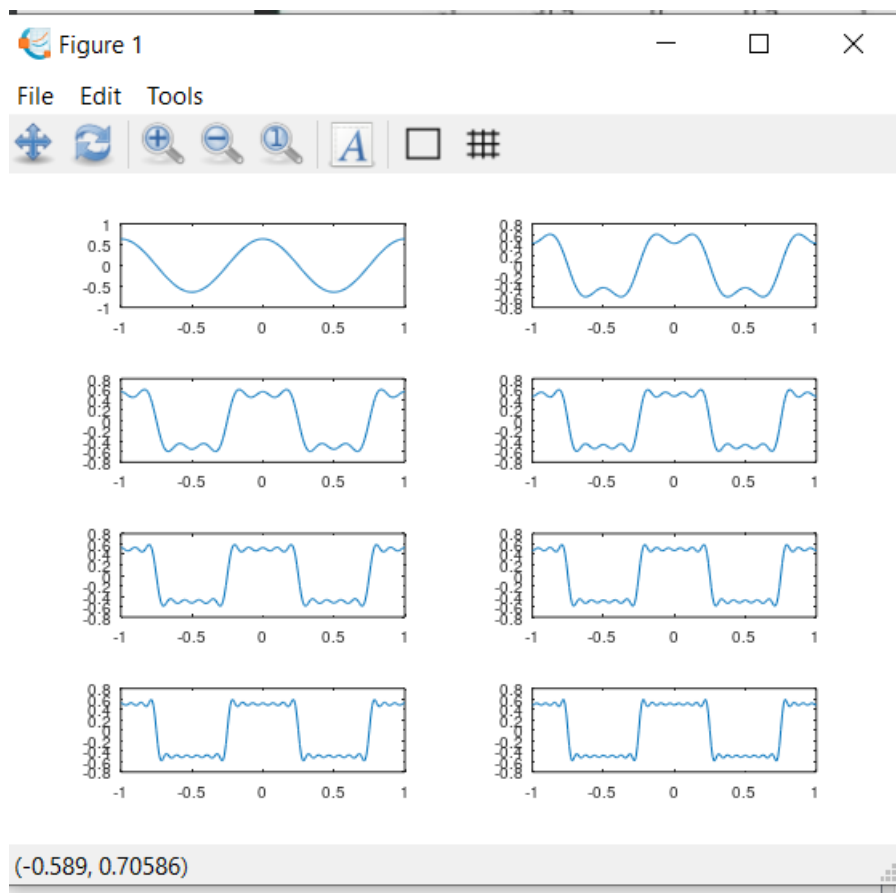
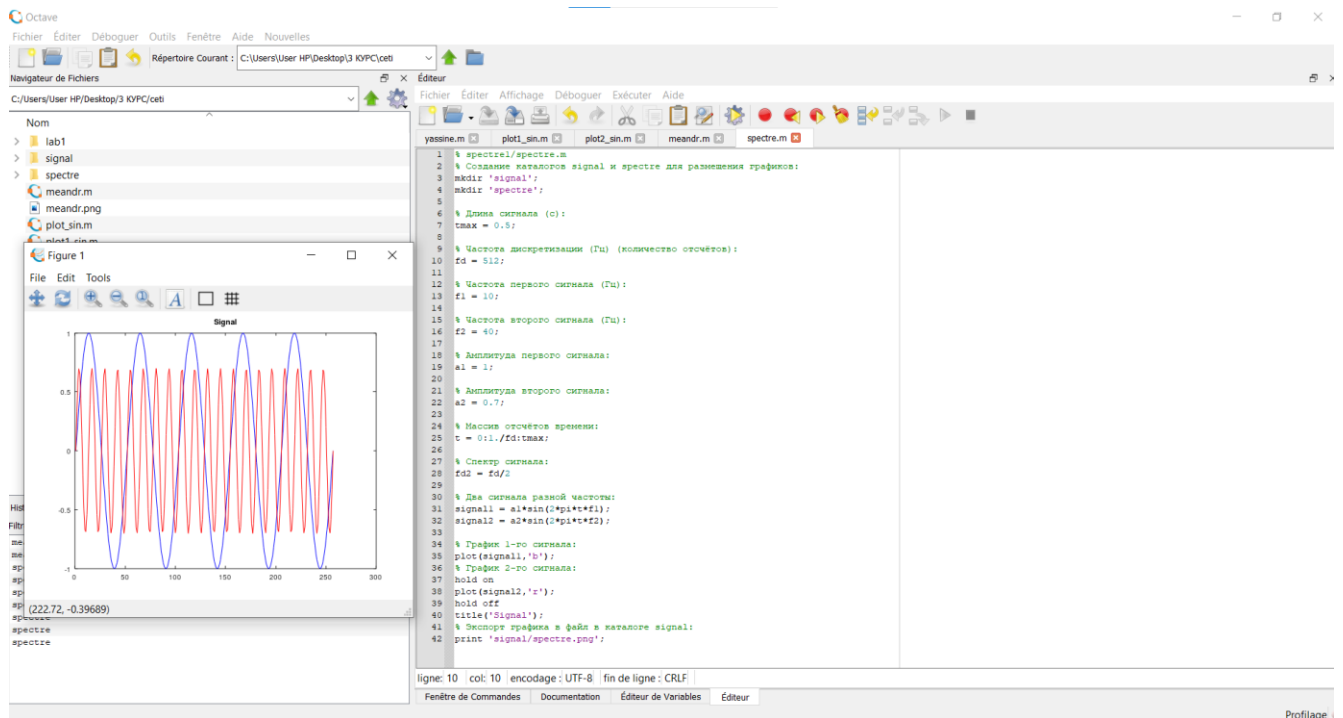


Рис. 1.3. Графики меандра, содержащего различное число гармоник



## 1.3.2.2. Порядок выполнения работы

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



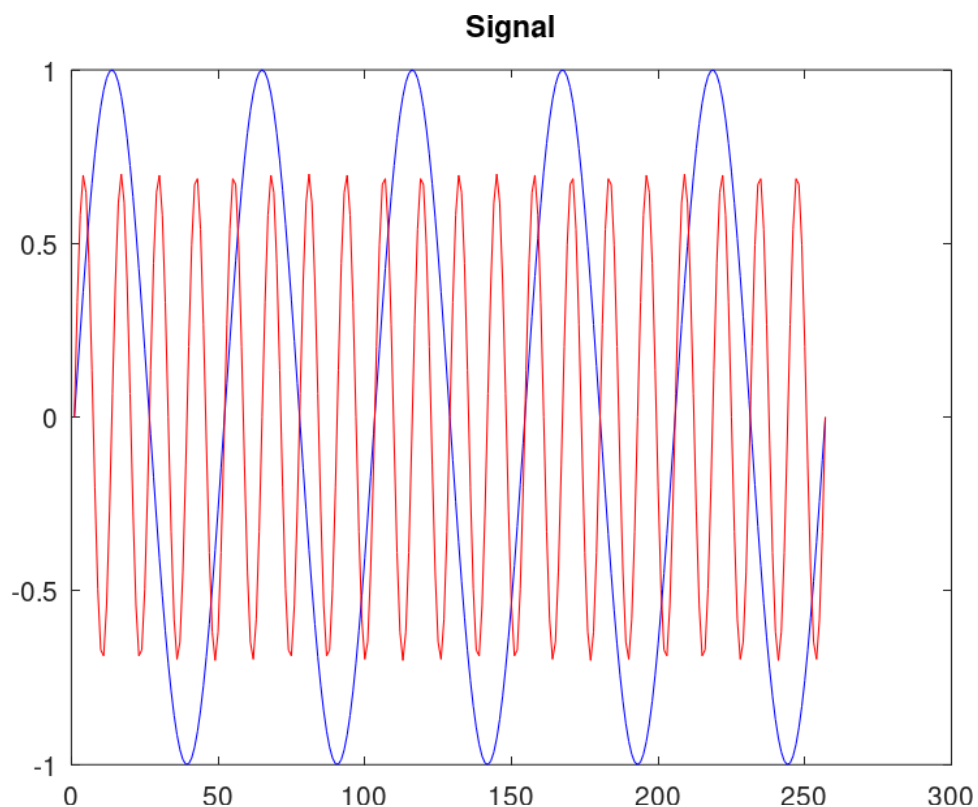


Рис. 1.4. Два синусоидальных сигнала разной частоты

**5. С помощью быстрого преобразования Фурье найдите спектры сигналов (рис. 1.5), добавив в файл `spectre.m` следующий код:**

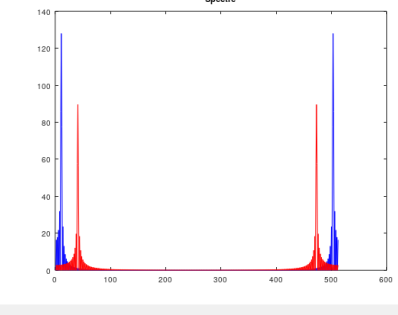
**6. Учитывая реализацию преобразования Фурье, скорректируйте график спектра (рис. 1.6): отбросьте дублирующие отрицательные частоты, а также примите в расчёт то, что на каждом шаге вычисления быстрого преобразования Фурье происходит суммирование амплитуд сигналов. Для этого добавьте в файл `spectre.m` следующий код:**



Fichier Éditer Débugger Outils Fenêtre Aide Nouvelles  
 Répertoire Courant : \\Users\User HP\Desktop\3 KYPC\cet\spectre

Nom  
 > signal  
 > spectre  
 > spectre.png  
 > spectre2.m

Figure 1  
 File Edit Tools  
 Spectre



```

13 f1 = 10;
14
15 % Частота второго сигнала (Гц):
16 f2 = 40;
17
18 % Амплитуда первого сигнала:
19 a1 = 1;
20
21 % Амплитуда второго сигнала:
22 a2 = 0.7;
23
24 % Массив отсчетов времени:
25 t = 0:1./fd:tmax;
26
27 % Спектр сигнала:
28 fd = fd/2;
29
30 % Два сигнала разной частоты:
31 signal1 = a1*sin(2*pi*t*f1);
32 signal2 = a2*sin(2*pi*t*f2);
33
34 % График 1-го сигнала:
35 plot(signal1,'b');
36 % График 2-го сигнала:
37 hold on
38 plot(signal2,'r');
39 hold off
40 title('Signal');
41 % Экспорт графика в файл в каталоге signal:
42 print 'signal/spectre.png';
43
44 % Посчитаем спектр
45 % Амплитуд преобразования Фурье сигнала 1:
46 spectre1 = abs(fft(signal1,fd));
47 % Амплитуд преобразования Фурье сигнала 2:
48 spectre2 = abs(fft(signal2,fd));
49 % Построение графиков спектров сигналов:
50 plot(spectre1,'b');
51 hold on
52 plot(spectre2,'r');
53 hold off
54 title('Spectre');
55 print 'spectre/spectre.png';
  
```

ligne: 55 col: 29 encodage: UTF-8 fin de ligne: CRLF  
 Fenêtre de Commandes Documentation Éditeur de Variables Éditeur

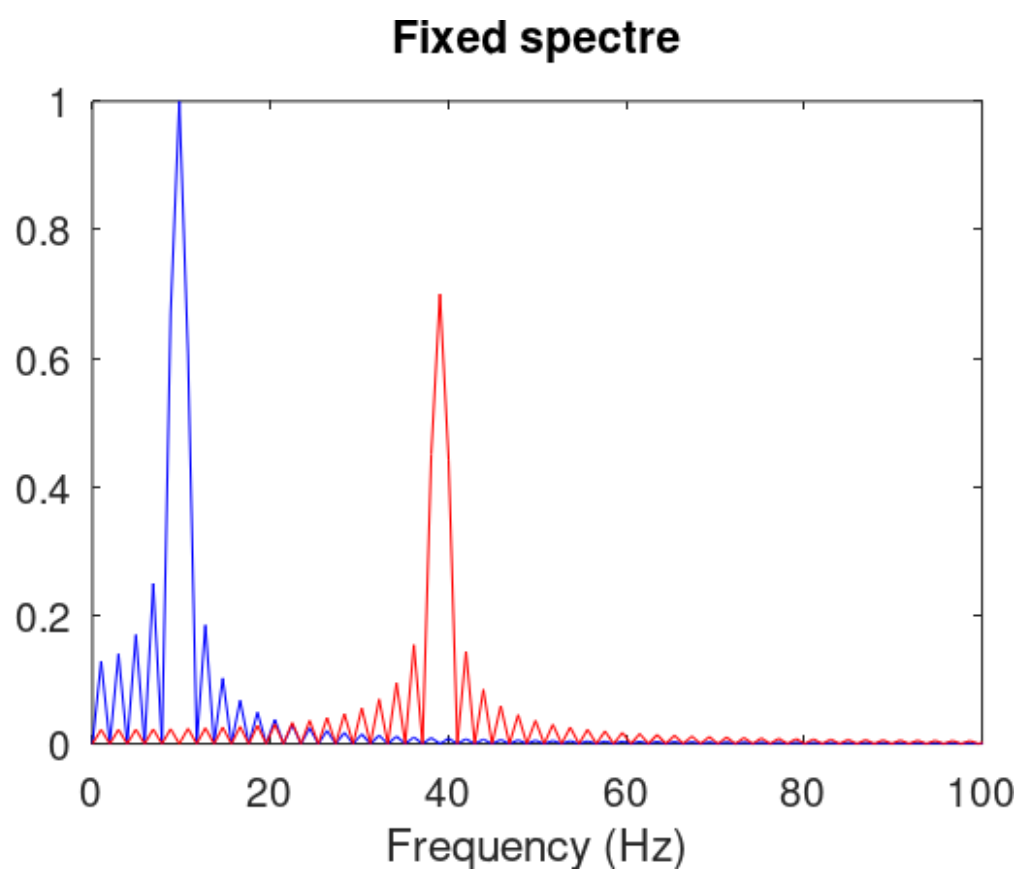


Рис. 1.6. Исправленный график спектров синусоидальных сигналов

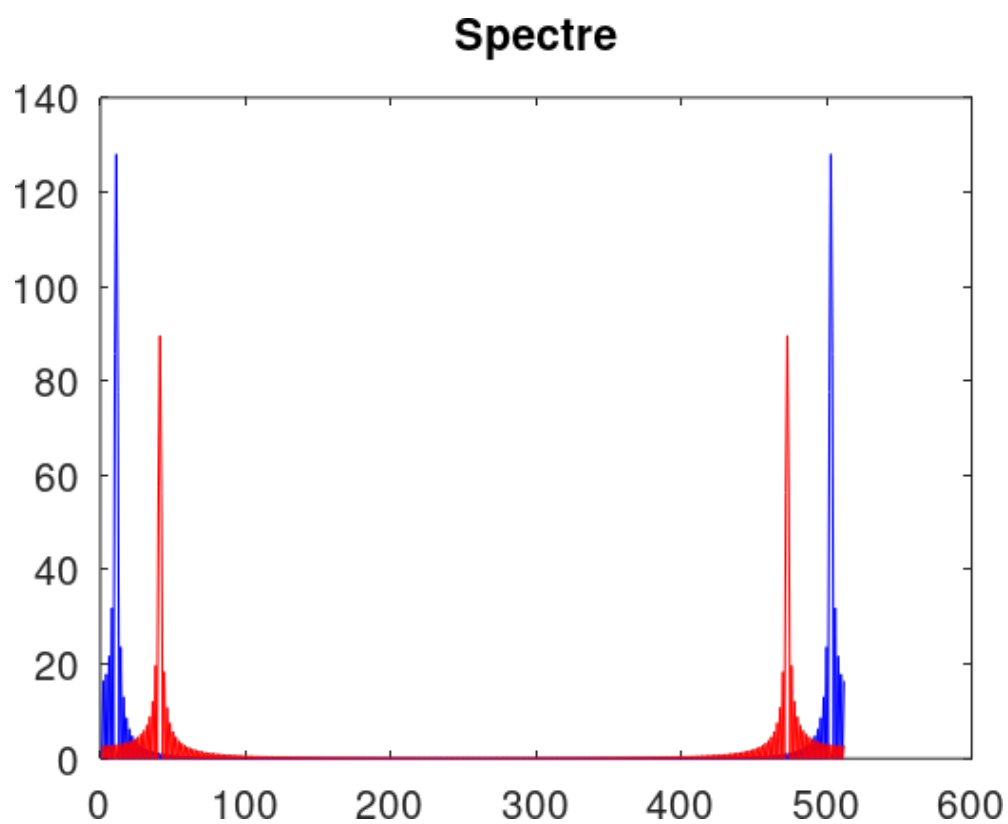
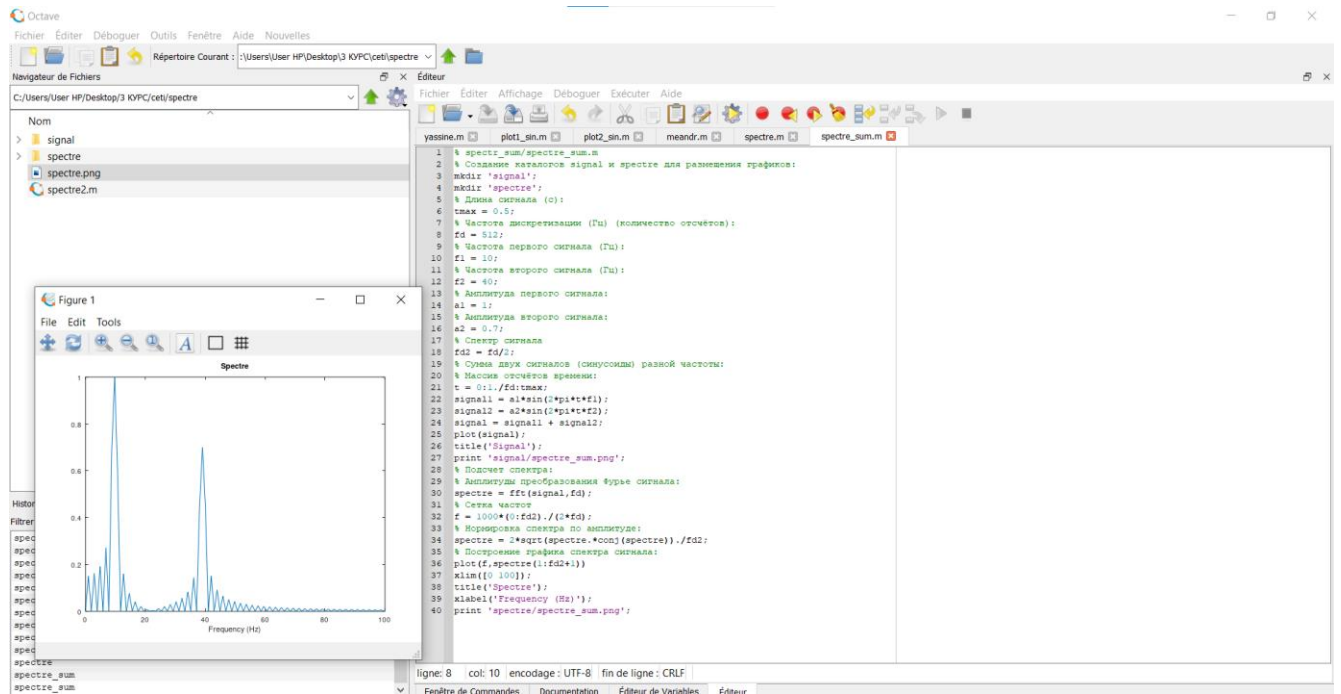


Рис. 1.5. График спектров синусоидальных сигналов

**7. Найдите спектр суммы рассмотренных сигналов (рис. 1.7), создав каталог spectr\_sum и файл в нём spectre\_sum.m со следующим кодом:**



The screenshot shows the Octave environment. On the left, a file explorer displays the directory structure: signal, spectre, spectre.png, and spectre2.m. In the center, a plot window titled 'Figure 1' shows a spectrum plot with 'Spectre' on the y-axis and 'Frequency (Hz)' on the x-axis. The plot shows two distinct peaks, one at a lower frequency and one at a higher frequency. On the right, a script editor shows the code for spectre\_sum.m. The code defines parameters for two signals, calculates their sum, and plots the resulting spectrum. The script includes comments in Russian and uses Octave functions like mkdir, fft, and plot.

```

1 % spectre_sum/spectre_sum.m
2 % Создание каталогов signal и spectre для размещения графиков:
3 mkdir 'signal';
4 mkdir 'spectre';
5 % Длина сигнала (s):
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 % Настройка 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:
29 % Амплитуды преобразования Фурье сигнала:
30 spectre = fft(signal,fd);
31 % Сумма частот
32 f = 1000*(0:fd2)/(1*fd);
33 % Нормировка спектра по амплитуде:
34 spectre = 2*sqrt(spectre.*conj(spectre))./fd2;
35 % Сохранение 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';

```



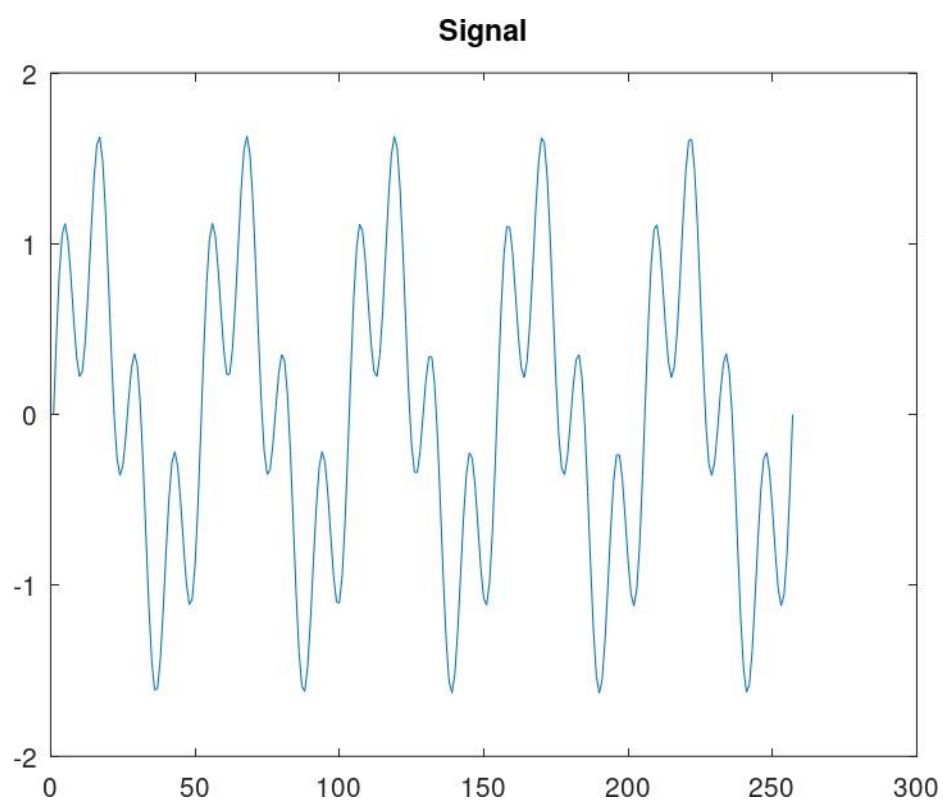


Рис. 1.7. Суммарный сигнал

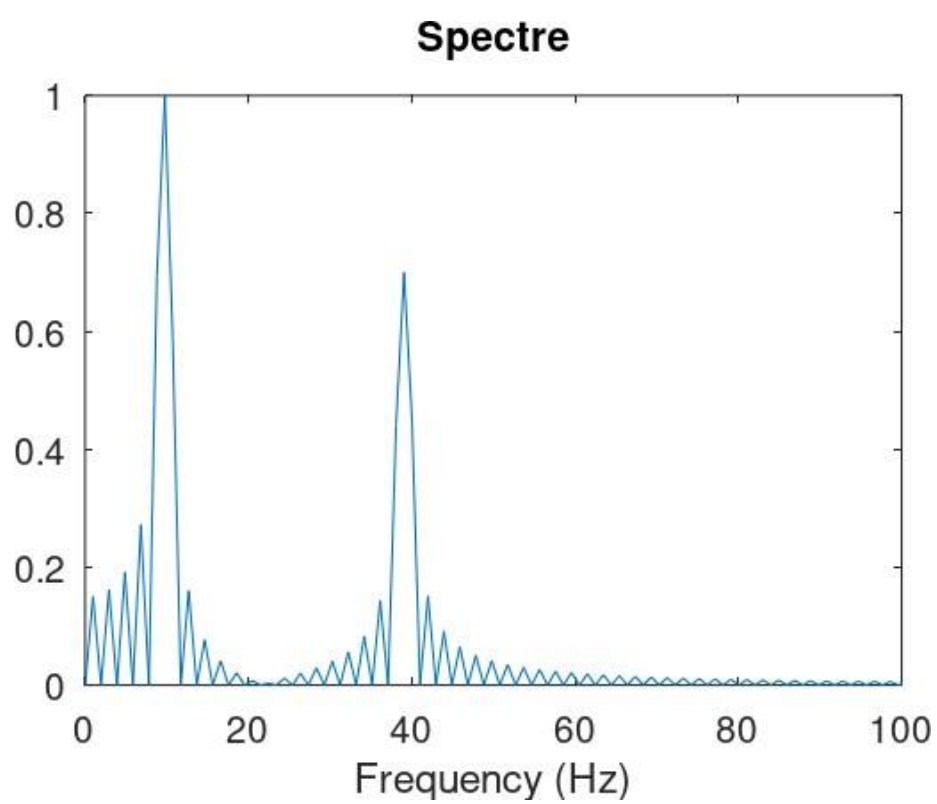


Рис. 1.8. Спектр суммарного сигнала

## 1.3.4. Амплитудная модуляция

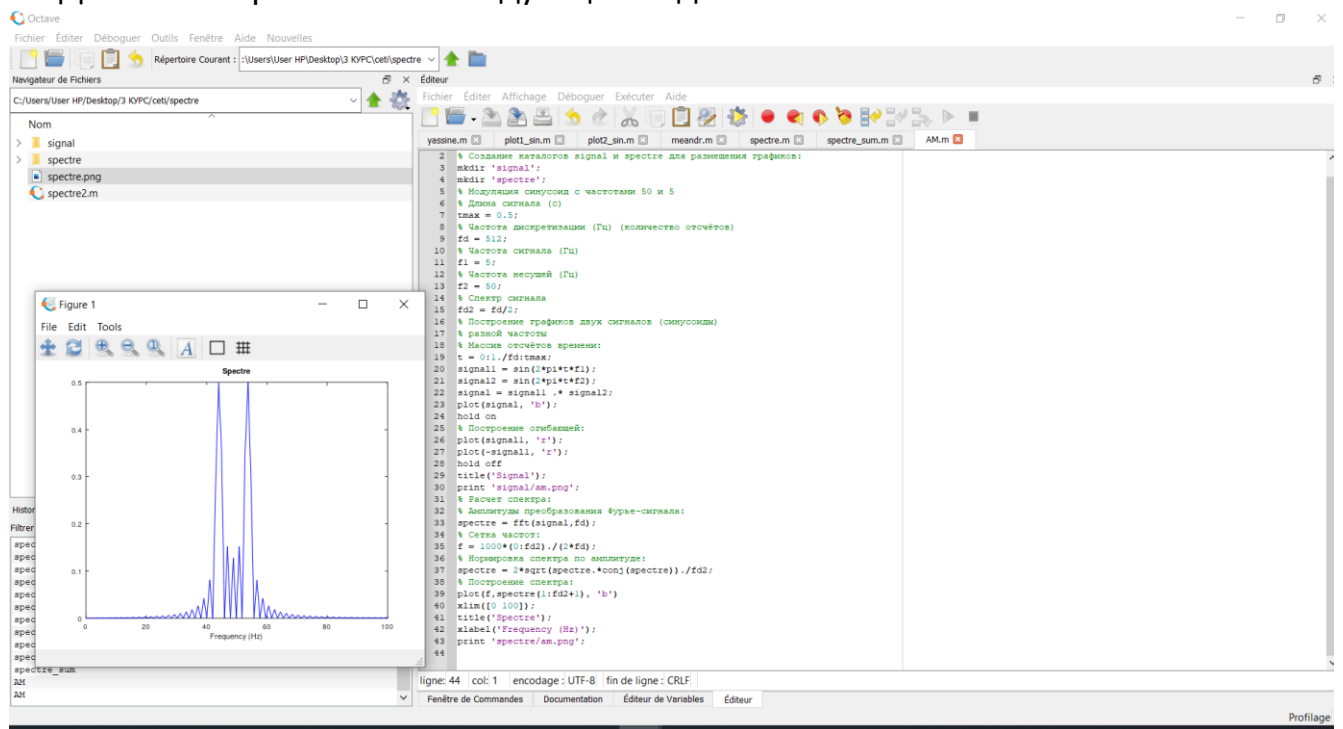
### 1.3.4.1. Постановка задачи

Продемонстрировать принципы модуляции сигнала на примере аналоговой амплитудной модуляции (рис.1.9).

### 1.3.4.2. Порядок выполнения работы

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

2. Добавьте в файле am.m следующий код:



В результате получаем, что спектр произведения представляет собой свёртку спектров (рис. 1.10).

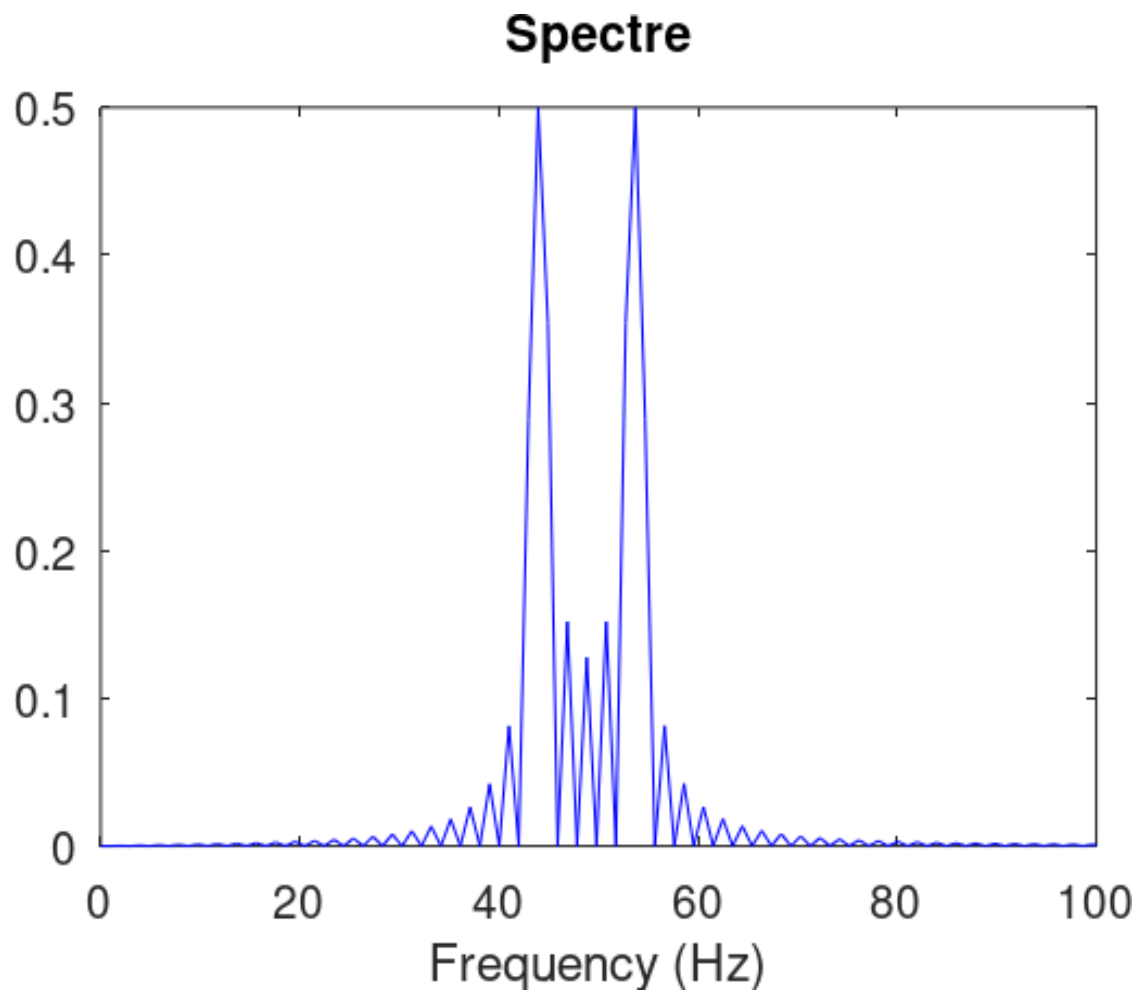


Рис. 1.10. Спектр сигнала при амплитудной модуляции

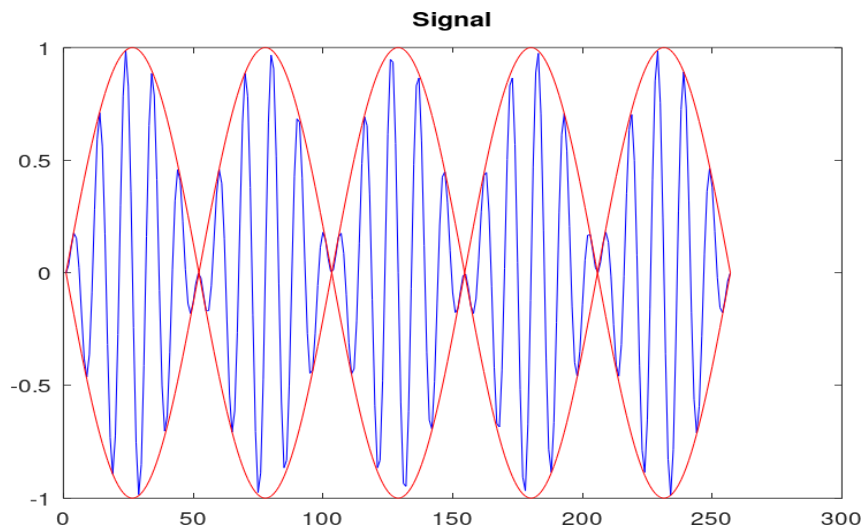


Рис. 1.9. Сигнал и огибающая при амплитудной модуляции

### 1.3.5. Кодирование сигнала. Исследование свойства самосинхронизации сигнала

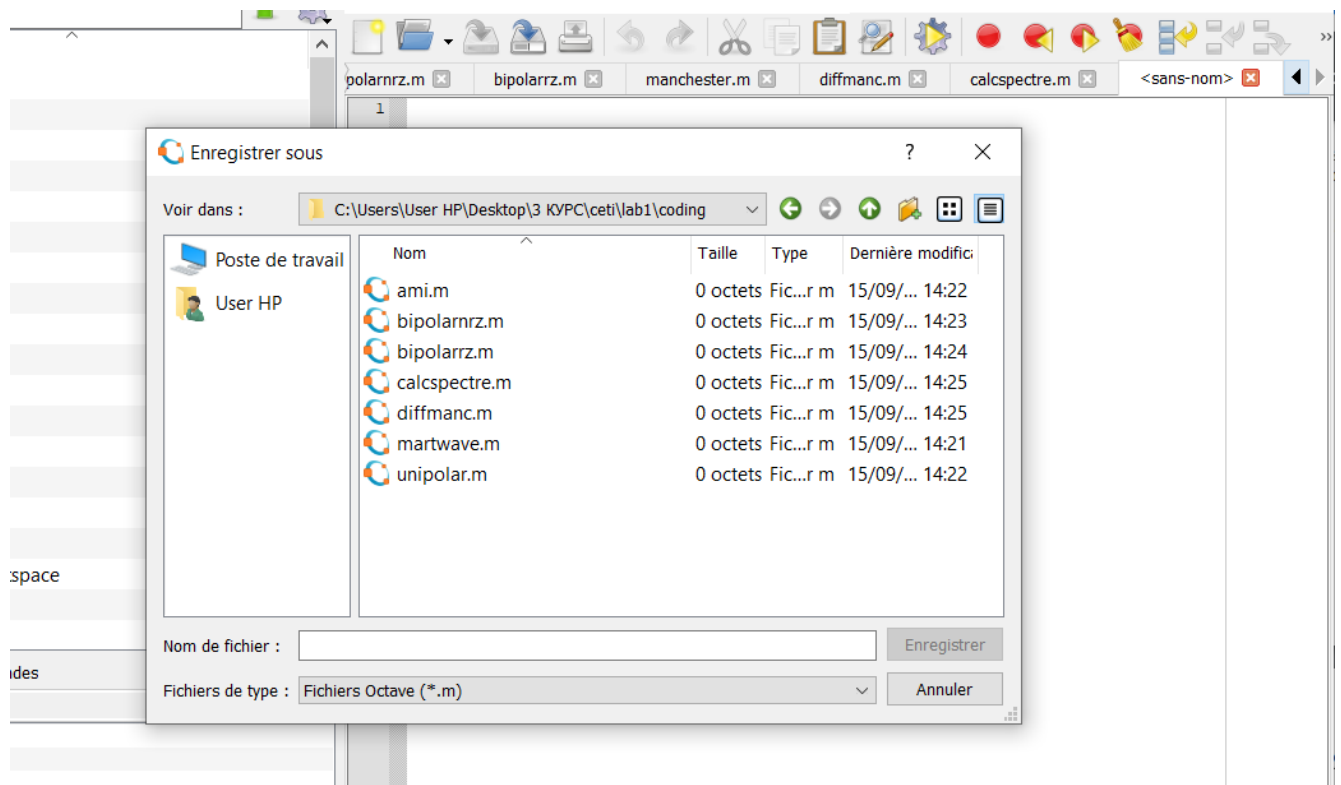
#### 1.3.5.1. Постановка задачи

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

#### 1.3.5.2. 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.





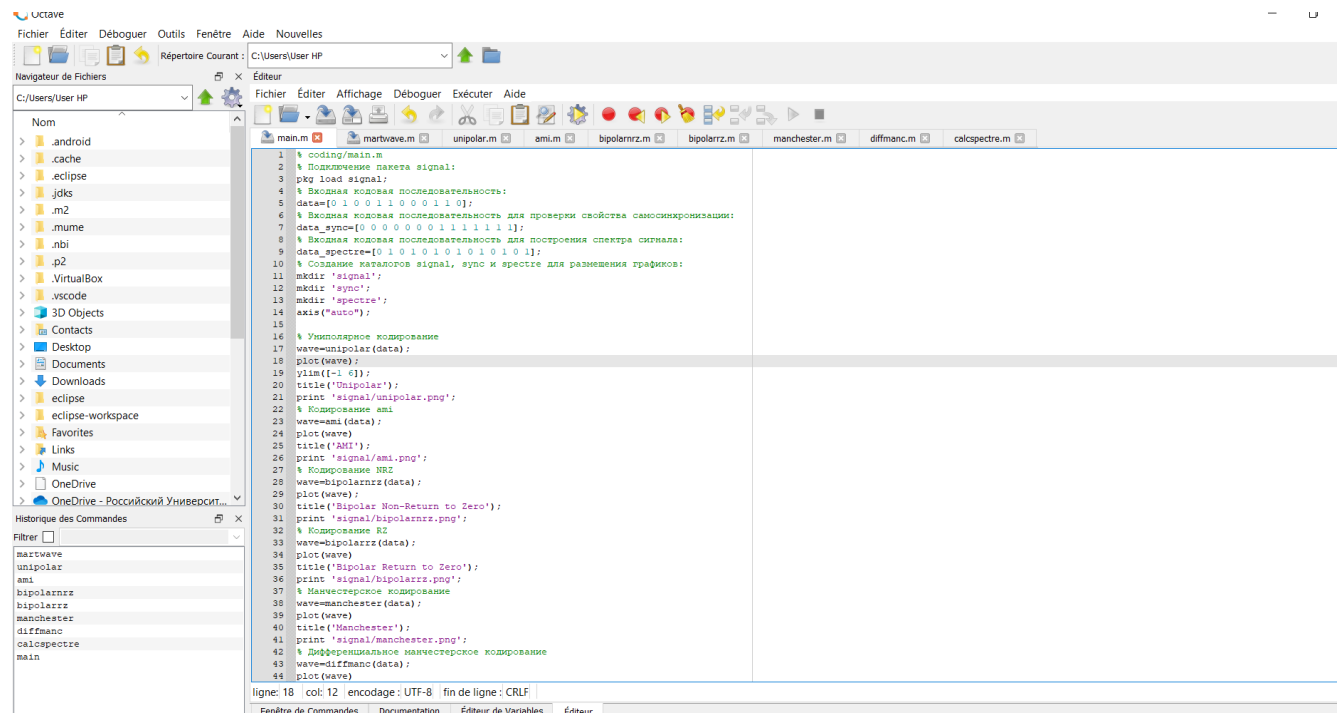
2. В окне интерпретатора команд проверьте, установлен ли у вас пакет расширений signal:

```

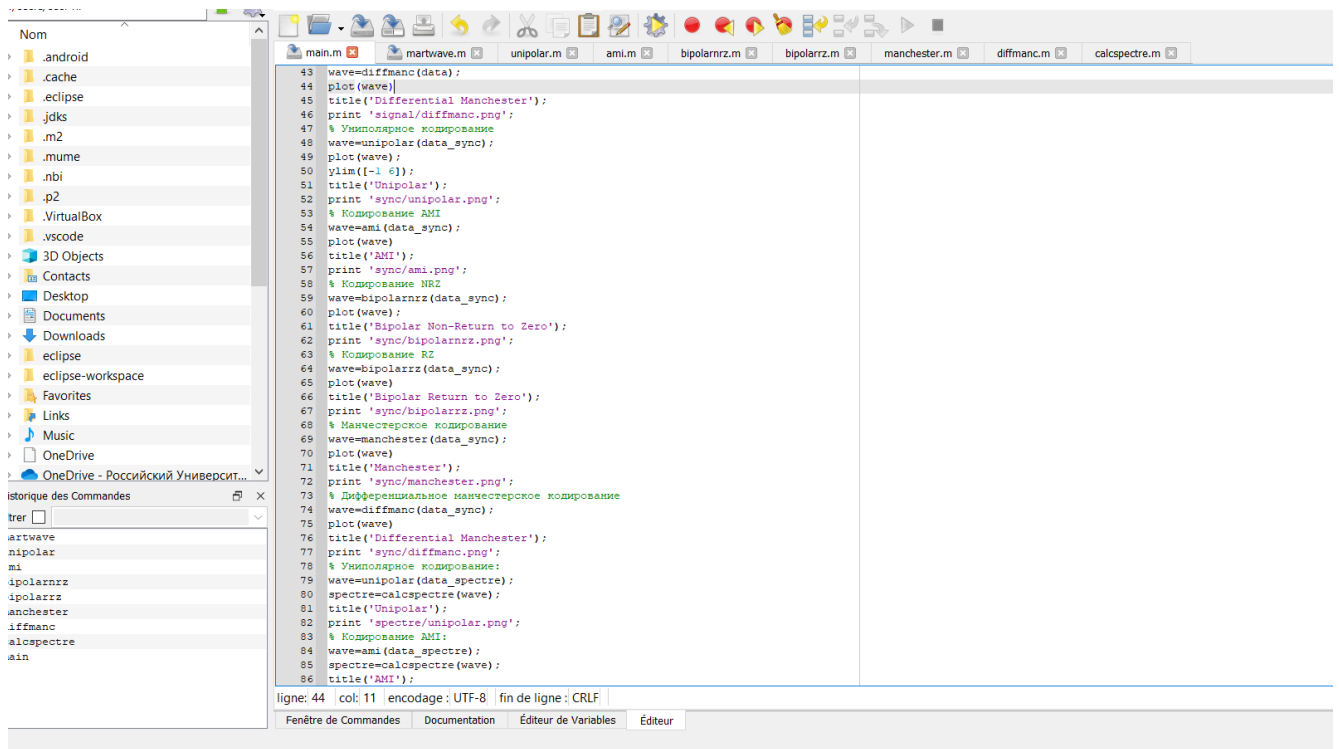
queueing | 1.2.7 | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\share\octave\packages\queueing-1.2.7
signal    | 1.4.2 | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\share\octave\packages\signal-1.4.2
sockets   | 1.4.0 | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\share\octave\packages\sockets-1.4.0

```

### 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 % Входная кодовая последовательность:
5 data=[0 1 0 0 1 1 0 0 0 1 1 0];
6 % Входная кодовая последовательность для проверки свойства самосинхронизации:
7 data_sync=[0 0 0 0 0 0 0 1 1 1 1 1];
8 % Входная кодовая последовательность для построения спектра сигнала:
9 data_spectre=[0 1 0 1 0 1 0 1 0 1 0 1];
10 % Создание каталогов signal, sync и spectre для размещения графиков:
11 mkdir 'signal';
12 mkdir 'sync';
13 mkdir 'spectre';
14 axis('auto');
15
16 % Униполярное кодирование
17 wave=unipolar(data);
18 plot(wave);
19 ylim([-1 6]);
20 title('Unipolar');
21 print 'signal/unipolar.png';
22 % Кодирование ami
23 wave=ami(data);
24 plot(wave);
25 title('AMI');
26 print 'signal/ami.png';
27 % Кодирование NRZ
28 wave=bipolarrr(data);
29 plot(wave);
30 title('Bipolar Non-Return to Zero');
31 print 'signal/bipolarrr.png';
32 % Кодирование RZ
33 wave=bipolarrrz(data);
34 plot(wave);
35 title('Bipolar Return to Zero');
36 print 'signal/bipolarrrz.png';
37 % Манчестерское кодирование
38 wave=manchester(data);
39 plot(wave);
40 title('Manchester');
41 print 'signal/manchester.png';
42 % Дифференциальное манчестерское кодирование
43 wave=diffmanc(data);
44 plot(wave);
```

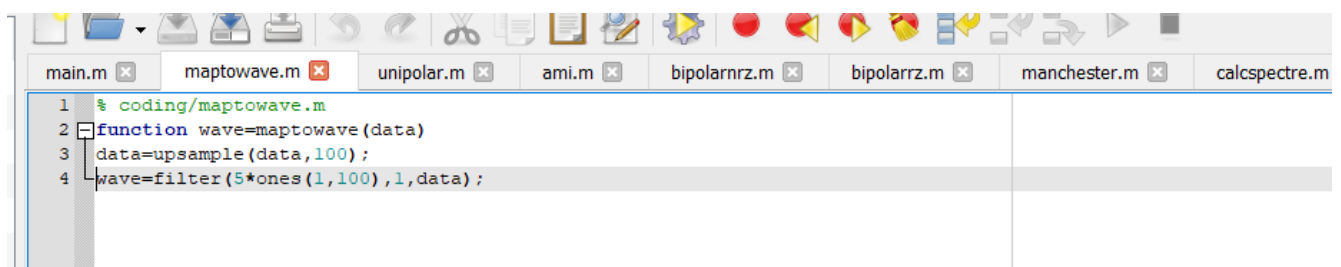


```

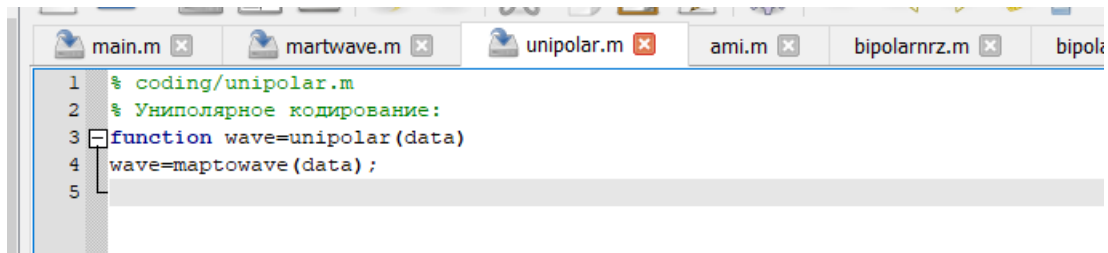
title('AMI');
print 'spectre/ami.png';
% Кодирование NRZ:
wave=bipolarnrz(data_spectre);
spectre=calcspectre(wave);
title('Bipolar Non-Return to Zero');
print 'spectre/bipolarnrz.png';
% Кодирование RZ:
wave=bipolarrz(data_spectre);
spectre=calcspectre(wave);
title('Bipolar Return to Zero');
print 'spectre/bipolarrz.png';
% 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';

```

**4. В файле maptowave.m пропишите функцию, которая по входному битовому потоку строит график сигнала:**



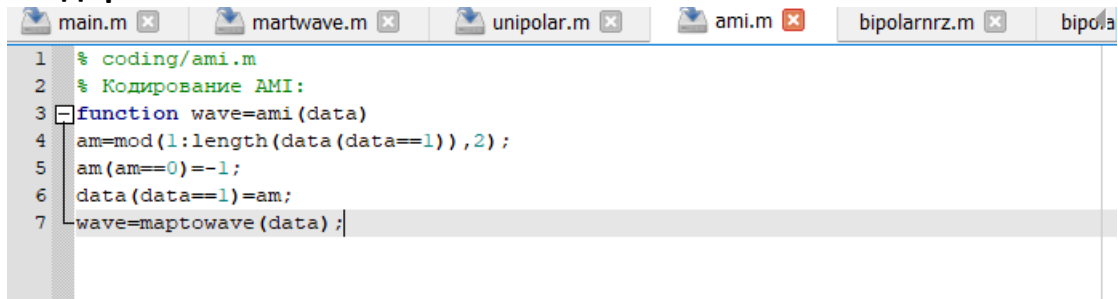
**5. В файлах unipolar.m, ami.m, bipolarnrz.m, bipolarrz.m, manchester.m, diffmanc.m пропишите соответствующие функции преобразования кодовой последовательности data с вызовом функции maptowave для построения соответствующего графика. Униполярное кодирование:**



The image shows a MATLAB editor window with several tabs: main.m, martwave.m, unipolar.m (active), ami.m, bipolarnrz.m, and bipola. The active file, unipolar.m, contains the following code:

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

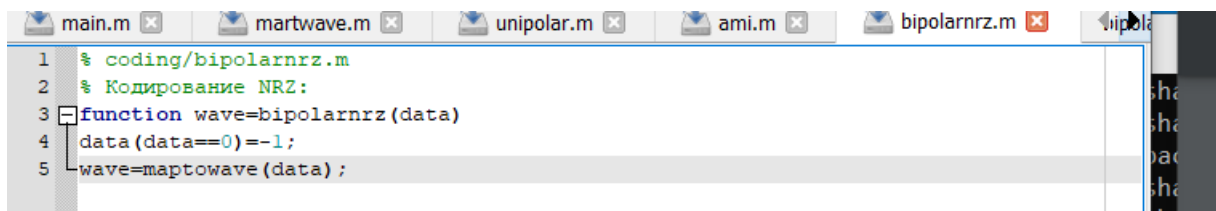
## Кодирование AMI:



The image shows a MATLAB editor window with tabs: main.m, martwave.m, unipolar.m, ami.m (active), bipolarnrz.m, and bipola. The active file, ami.m, contains the following code:

```
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);
```

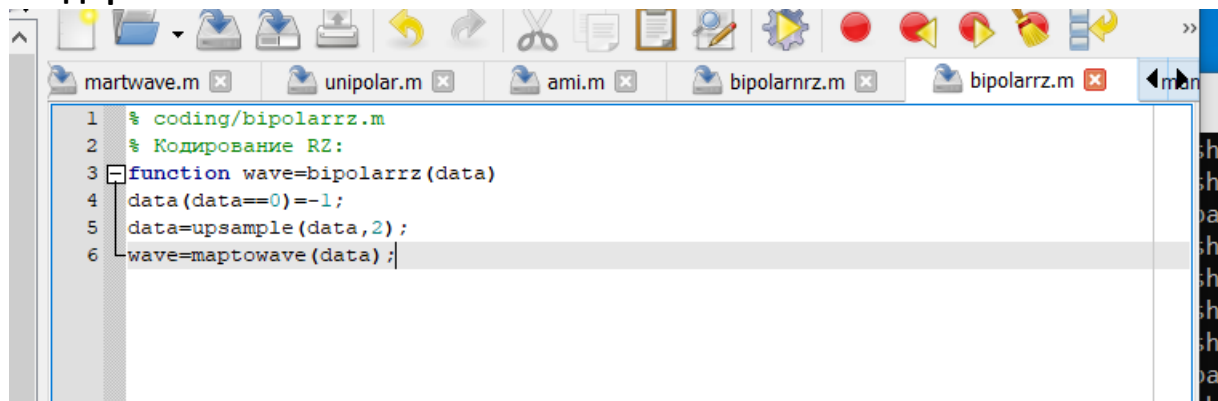
## Кодирование NRZ:



The image shows a MATLAB editor window with tabs: main.m, martwave.m, unipolar.m, ami.m, bipolarnrz.m (active), and bipola. The active file, bipolarnrz.m, 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);
```

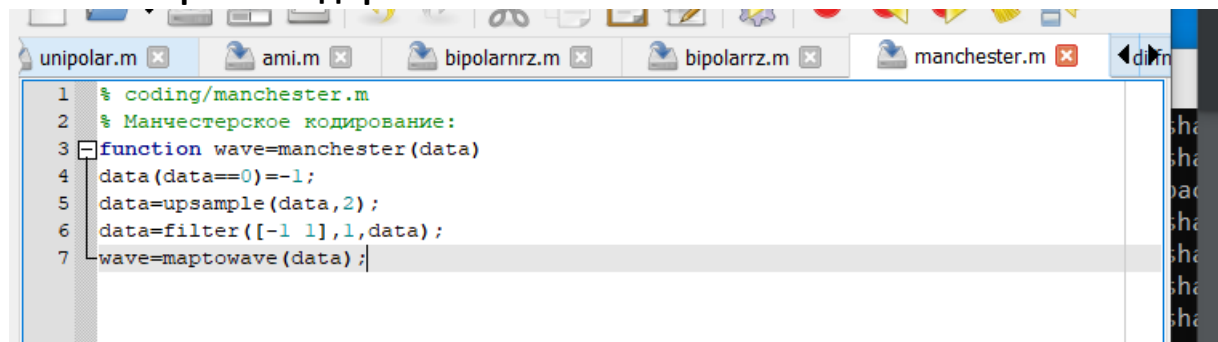
### Кодирование RZ:



The image shows a MATLAB editor window with the file 'bipolarrrz.m' open. The code defines a function 'wave=bipolarrrz(data)' that implements RZ encoding. It sets data to -1 for zero values, upsamples by 2, and then calls 'maptowave'.

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

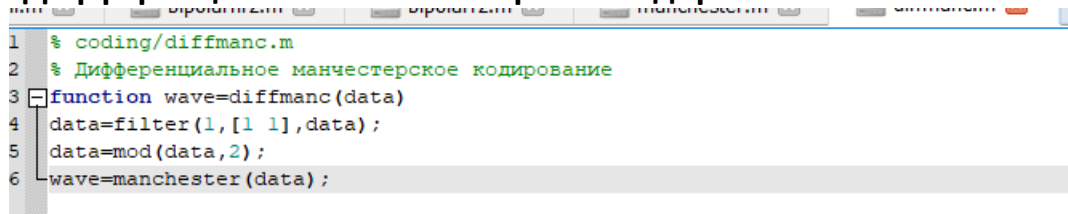
### Манчестерское кодирование:



The image shows a MATLAB editor window with the file 'manchester.m' open. The code defines a function 'wave=manchester(data)' that implements Manchester encoding. It sets data to -1 for zero values, upsamples by 2, filters with a [1 -1] filter, and then calls 'maptowave'.

```
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);
```

### Дифференциальное манчестерское кодирование:



The image shows a MATLAB editor window with the file 'diffmanc.m' open. The code defines a function 'wave=diffmanc(data)' that implements differential Manchester encoding. It filters the data with a [1 1] filter, takes the modulo 2, and then calls 'manchester'.

```
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);
```

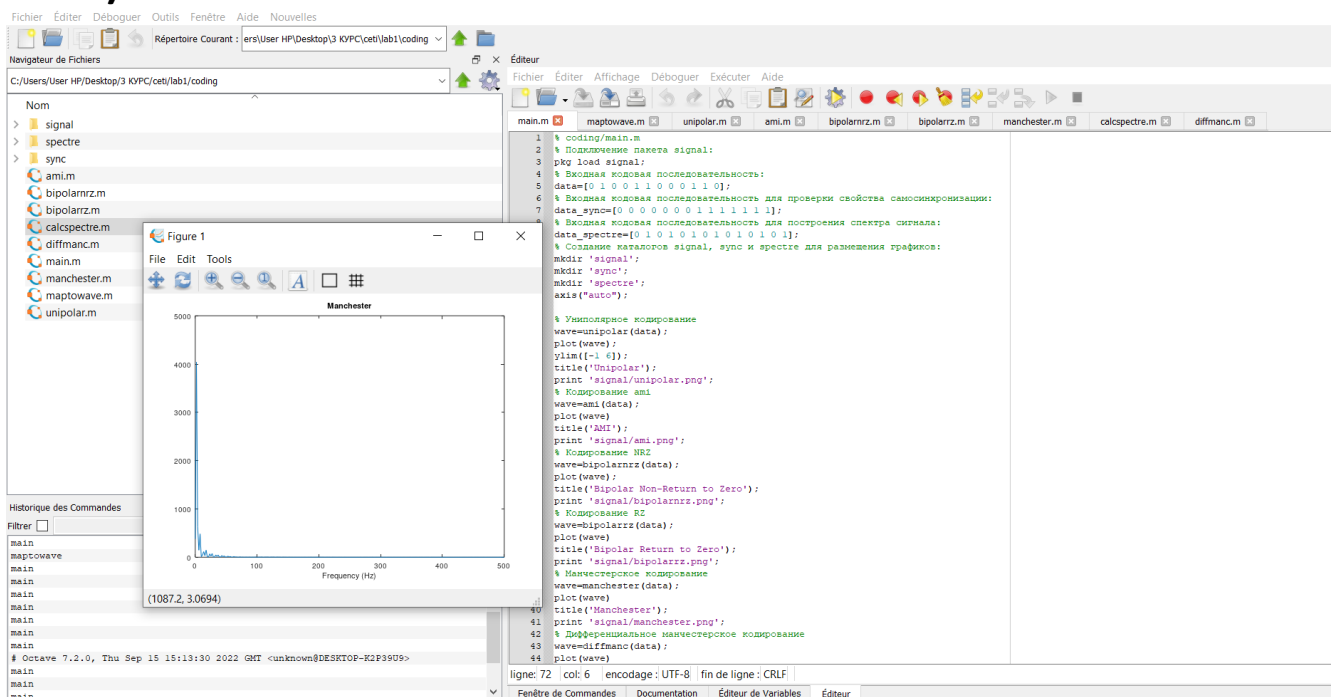
6. В файле calcspectre.m пропишите функцию построения спектра сигнала:

```

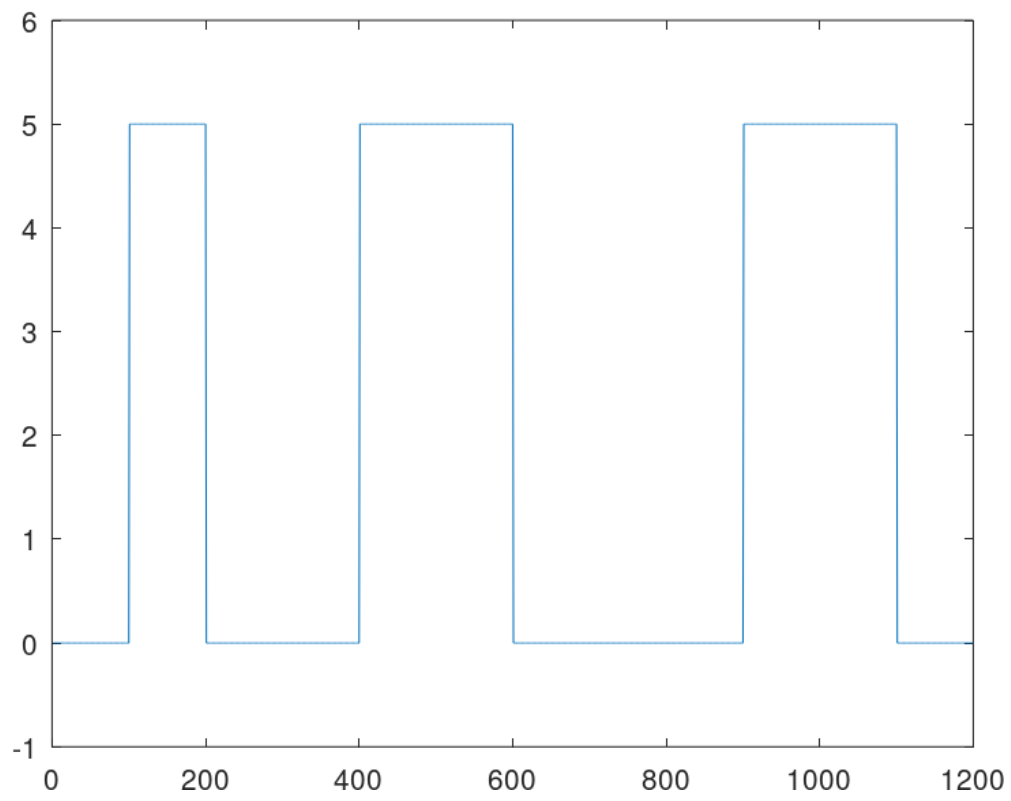
1 % calcspectre.m
2 % функция построения спектра сигнала:
3 function spectre = calcspectre(wave)
4     % Частота дискретизации (Гц):
5     Fd = 512;
6     Fd2 = Fd/2;
7     Fd3 = Fd/2 + 1;
8     X = fft(wave,Fd);
9     spectre = X.*conj(X)/Fd;
10    f = 1000*(0:Fd2)/Fd;
11    plot(f,spectre(1:Fd3));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

```

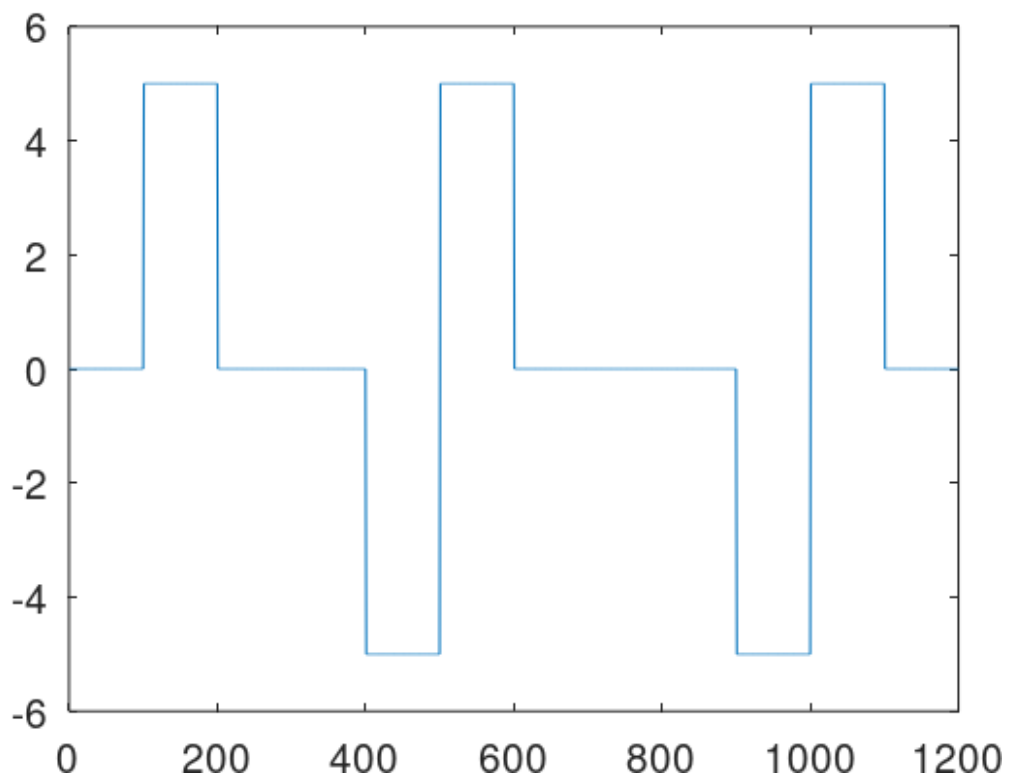
**7. Запустите главный скрипт main.m. В каталоге signal должны быть получены файлы с графиками кодированного сигнала (рис. 1.11–1.16), в каталоге sync — файлы с графиками, иллюстрирующими свойства самосинхронизации (рис. 1.17–1.22), в каталоге spectre — файлы с графиками спектров сигналов (рис. 1.23–1.28).**



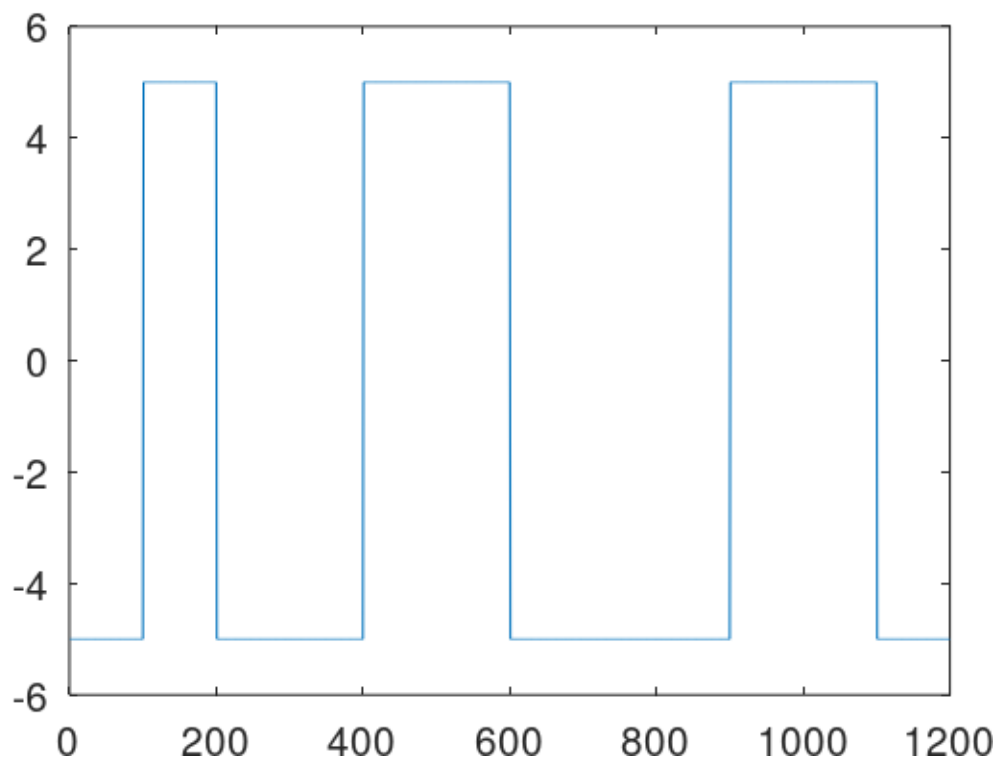
**Unipolar**



**AMI**

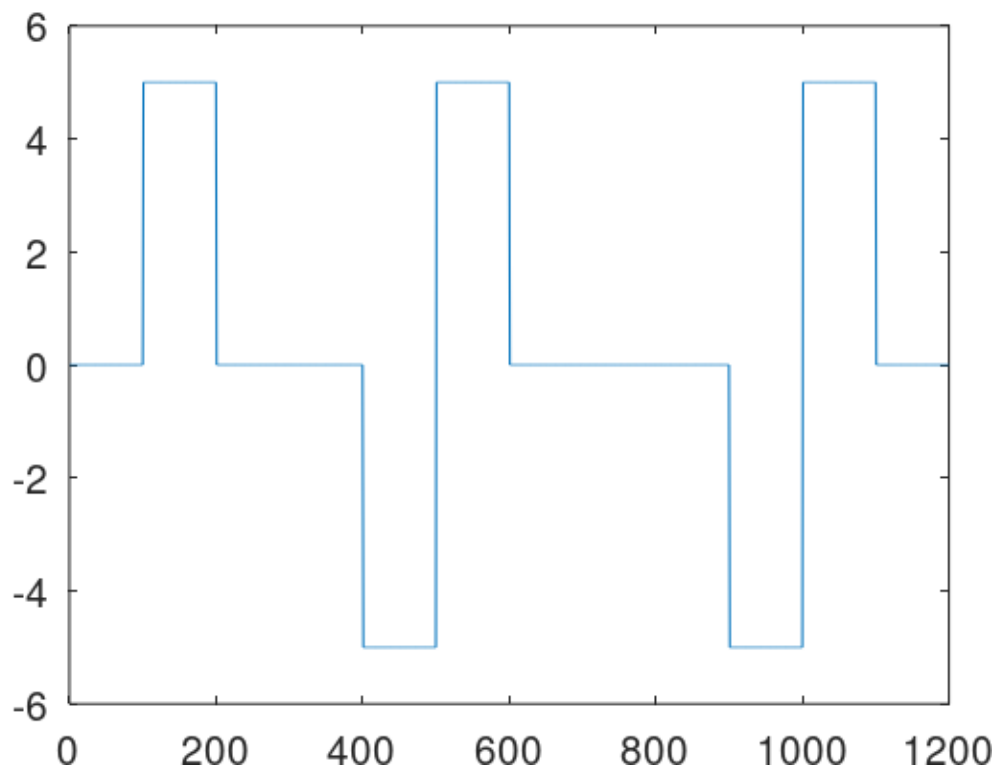


**Bipolar Non-Return to Zero**

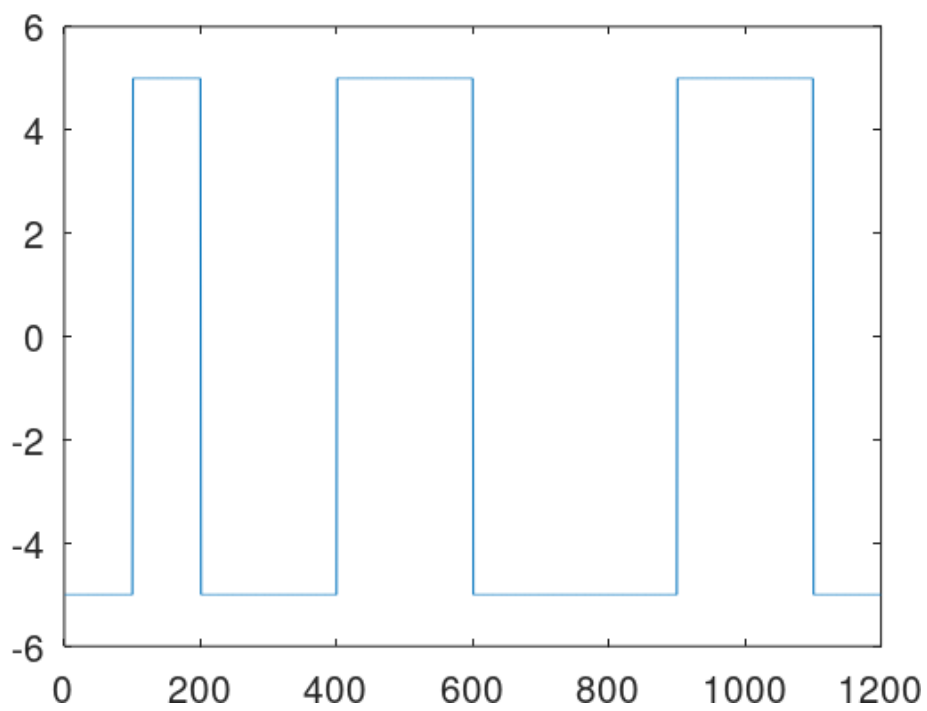




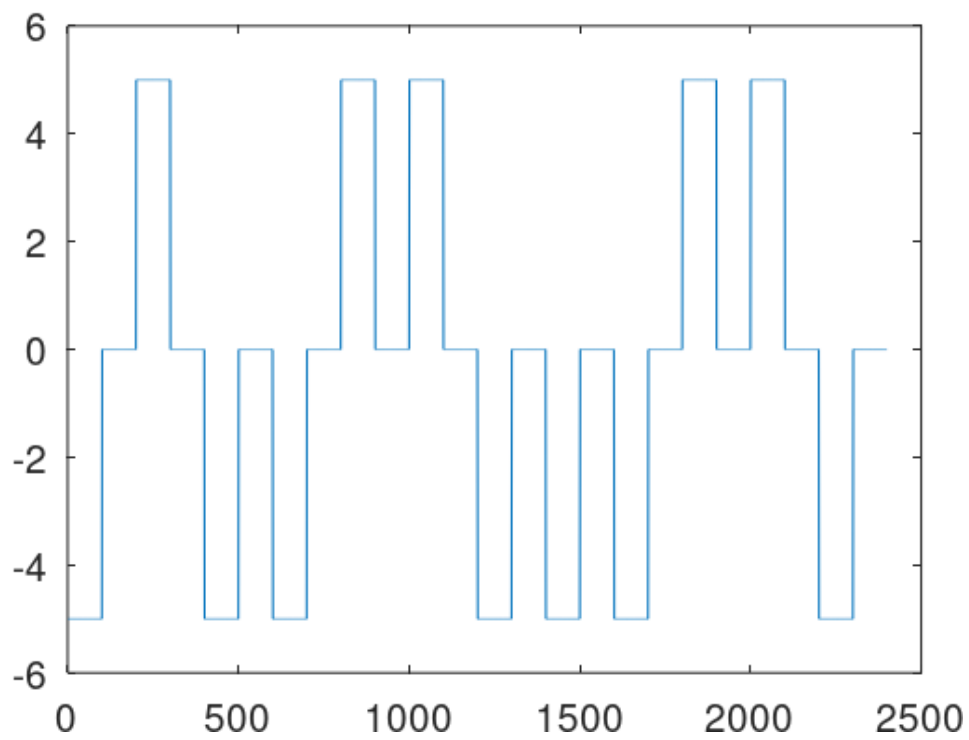
**AMI**



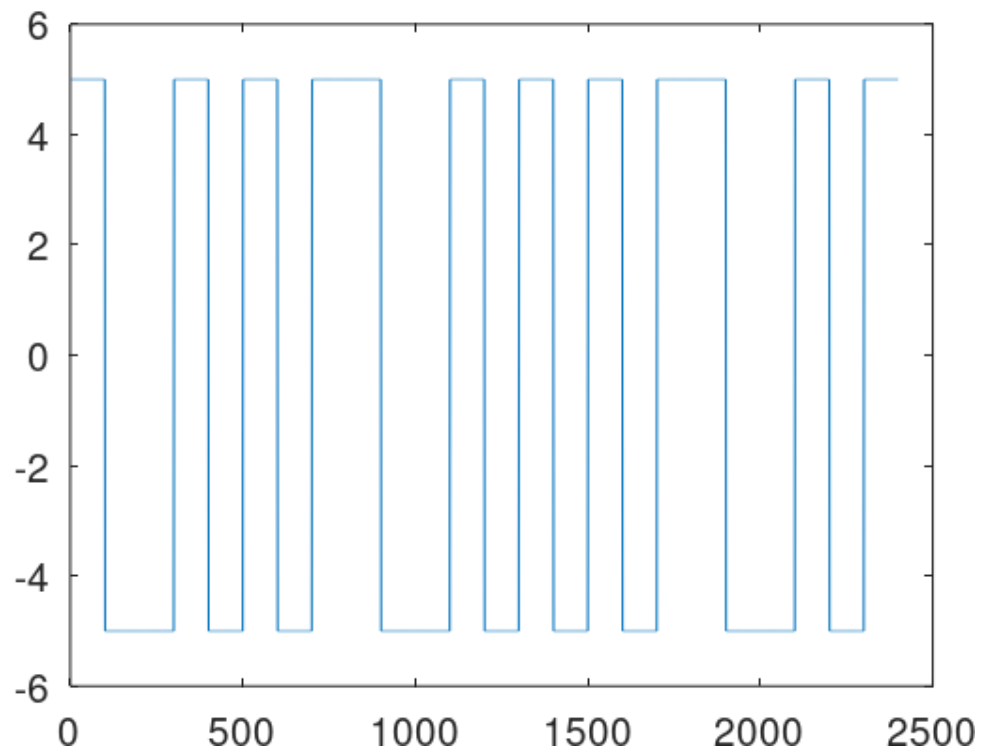
**Bipolar Non-Return to Zero**



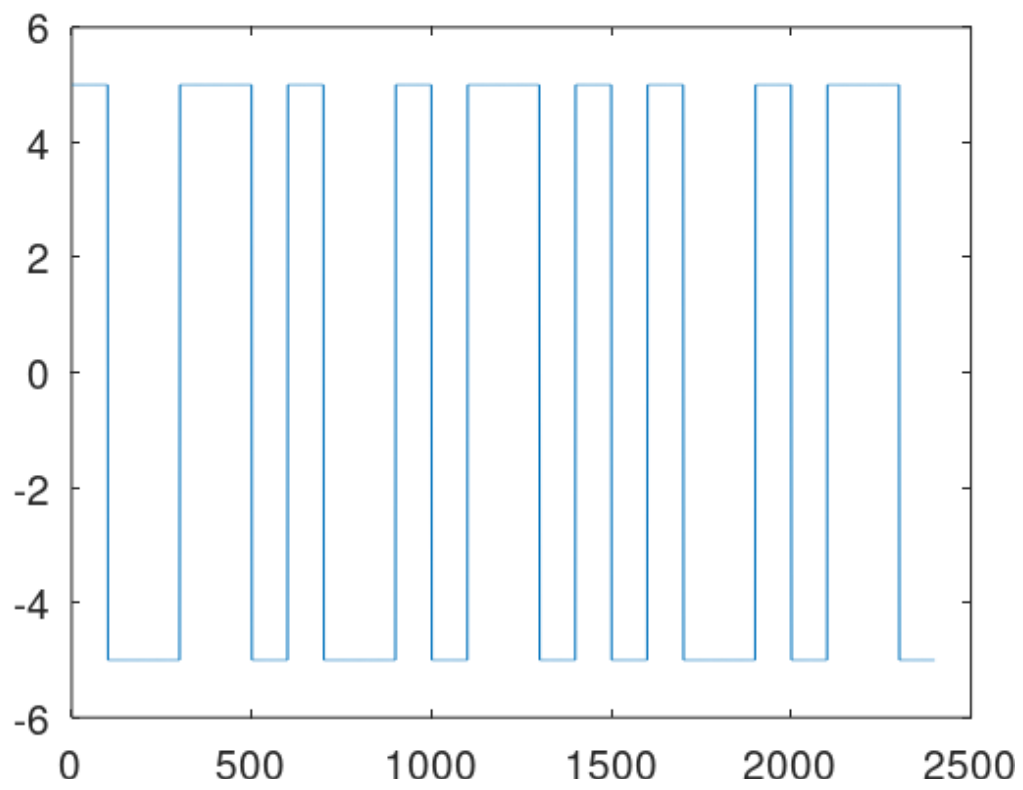
**Bipolar Return to Zero**



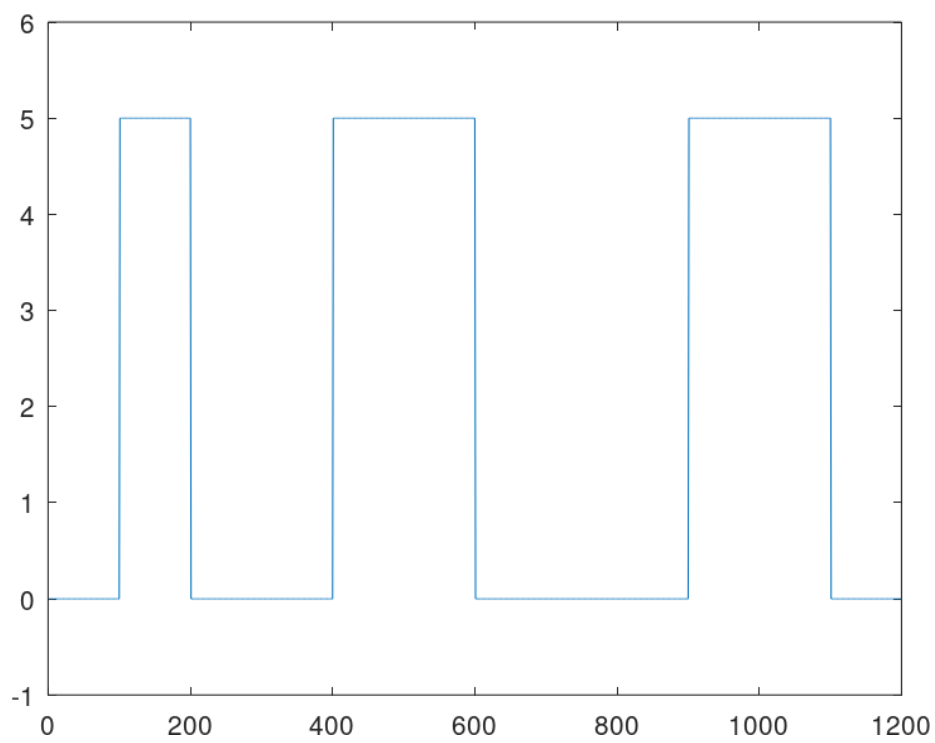
**Differential Manchester**



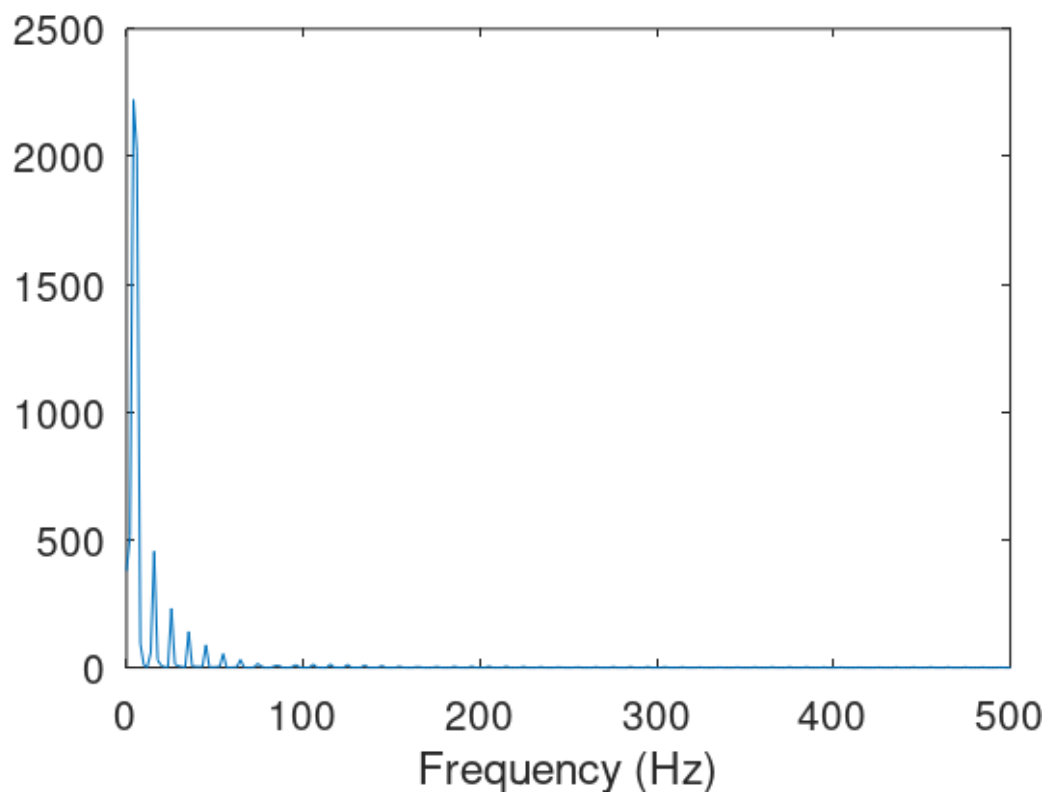
**Manchester**



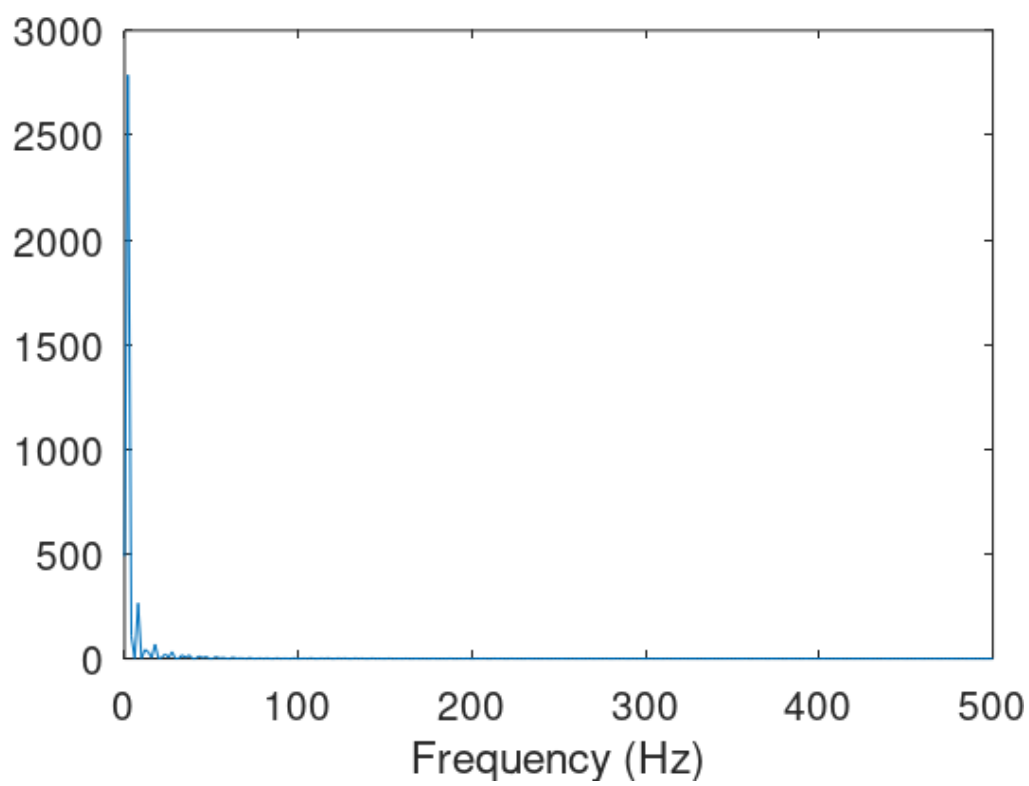
**Unipolar**



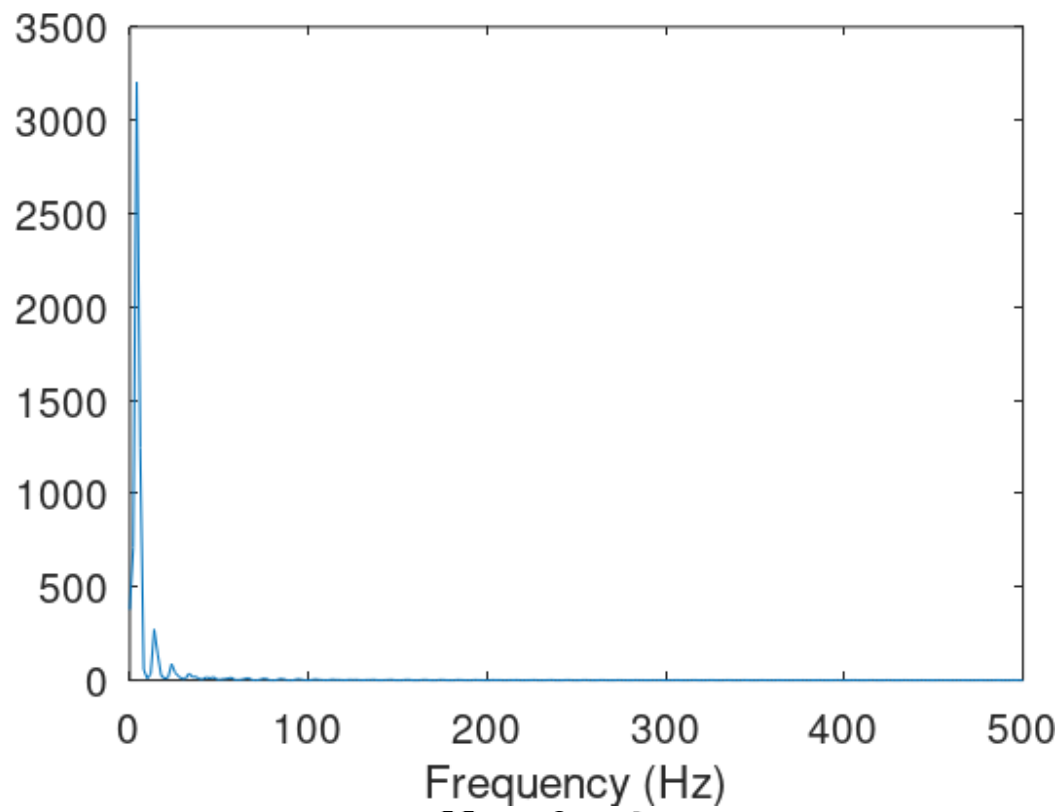
**Bipolar Non-Return to Zero**



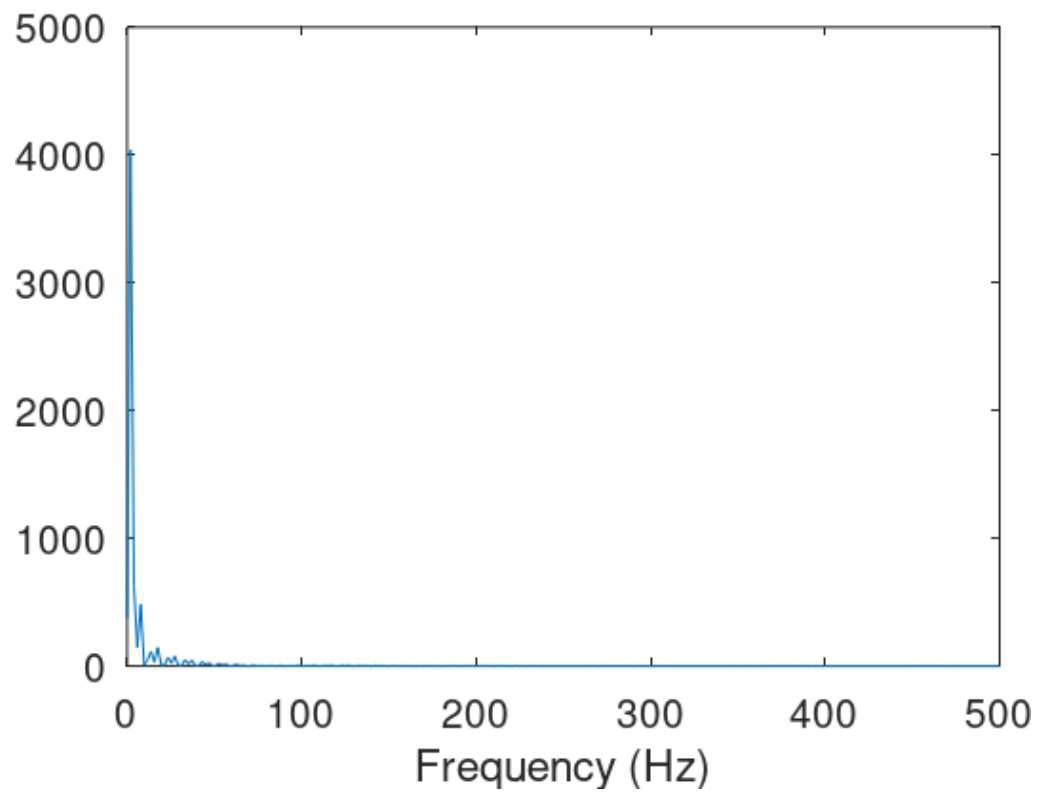
**Bipolar Return to Zero**



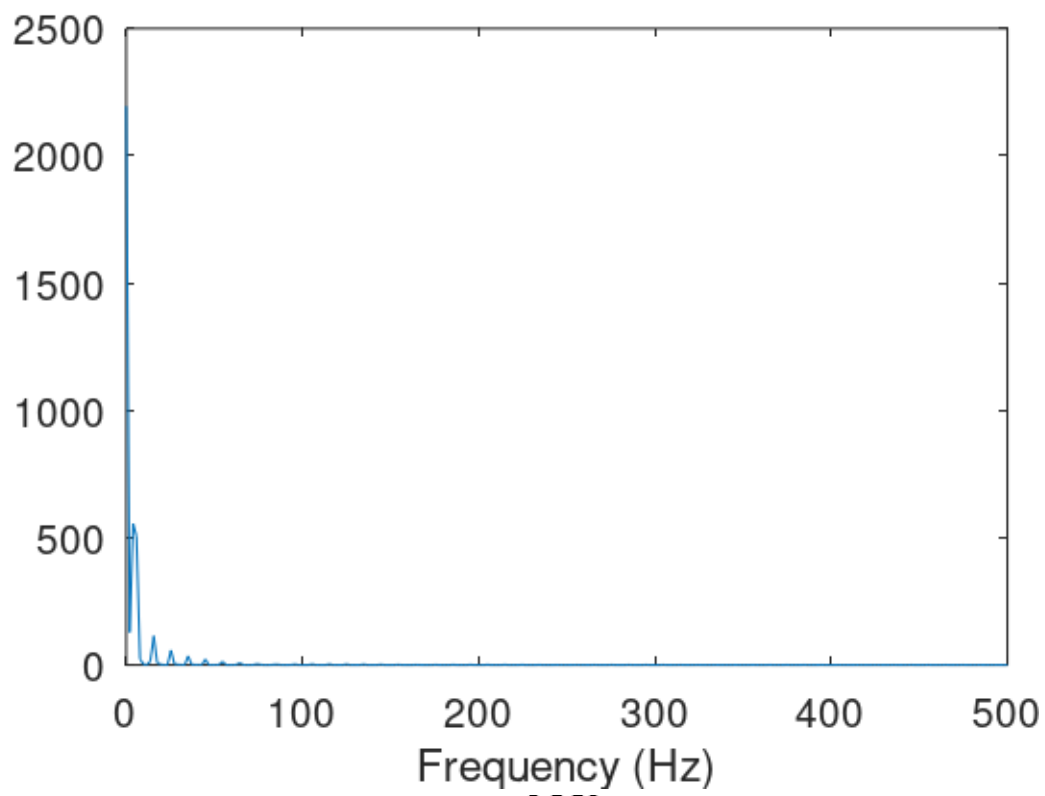
## Differential Manchester



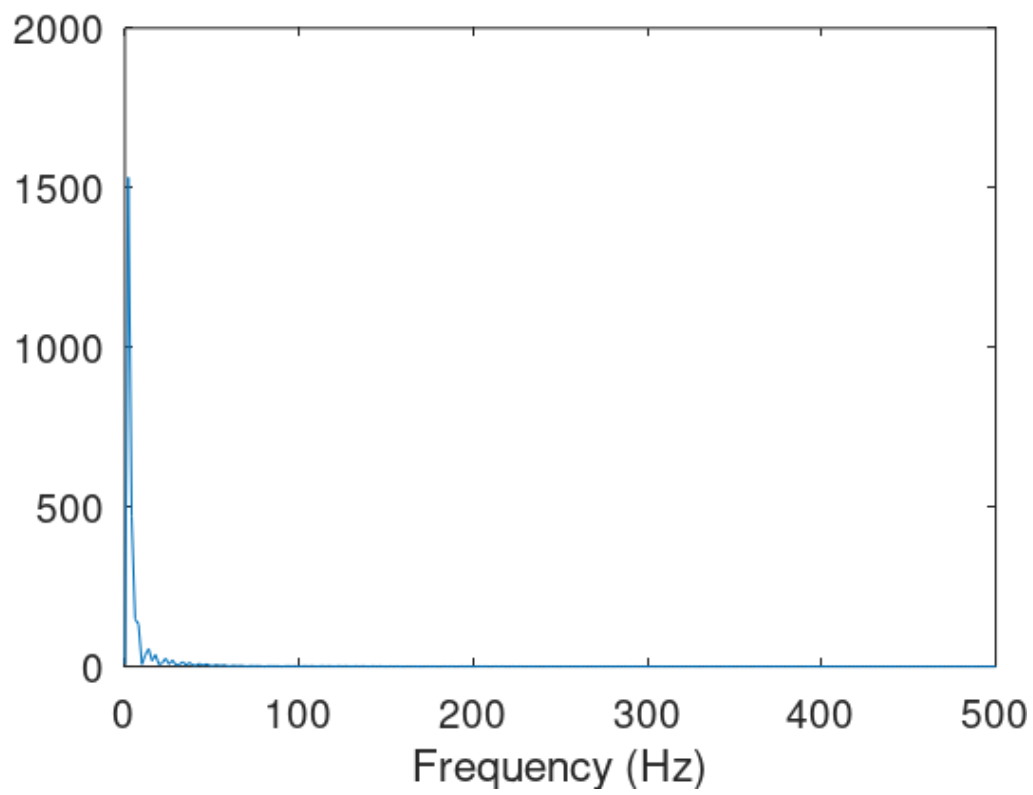
## Manchester

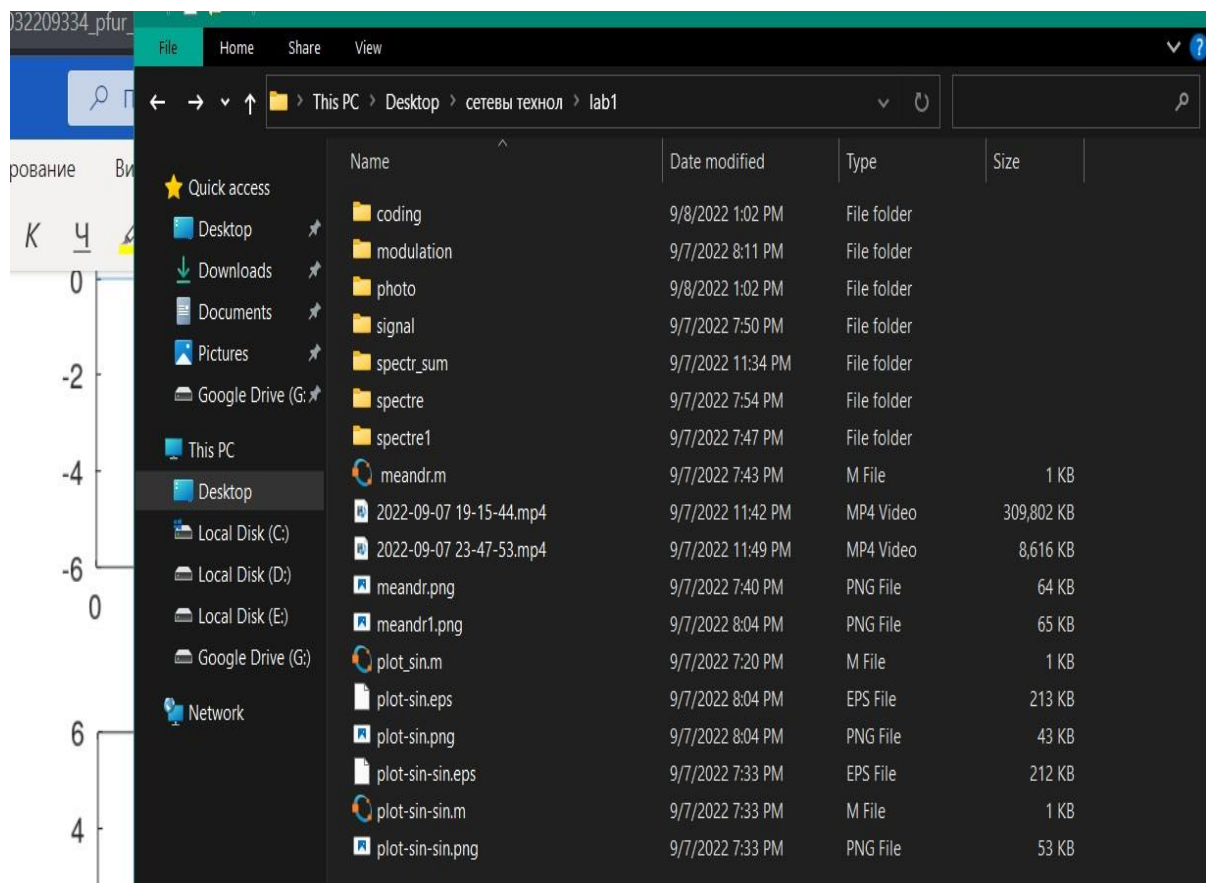


**Unipolar**



**AMI**





## Вывод

Было действительно интересно узнать об Октаве. Теперь я знаю, как продемонстрировать конкретные графики с помощью Октавы. Задания не были сложными, вам просто нужно внимательно прочитать инструкцию













