

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

Дисциплина «Сетевые технологии»

Тема «Методы кодирования и модуляция сигналов»

Студент: Щербак Маргарита Романовна

Ст. билет: 1032216537

Группа: НПИбд-02-21

МОСКВА

2023 г.

Цели работы

Изучить методы кодирования и модуляции сигналов с помощью высокоуровневого языка программирования Octave. Определить спектр и параметры сигнала. Продемонстрировать принципы модуляции сигнала на примере аналоговой амплитудной модуляции. Исследовать свойства самосинхронизации сигнала.

Выполнение работы

1. Построение графиков в Octave

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`.

1.2. Выполнение

1. Запустила Octave с оконным интерфейсом.
2. Перешла в окно редактора и создала новый сценарий с именем `plot_sin.m`, затем сохранила его в свой рабочий каталог (рис.1.1).

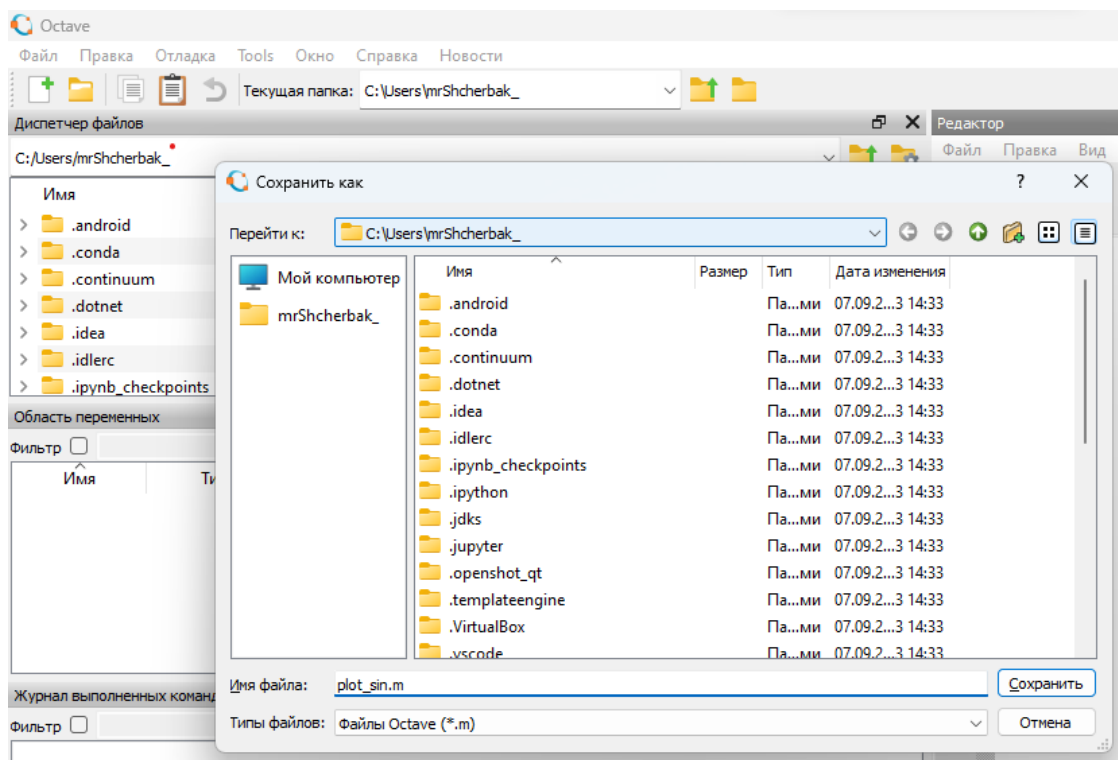


Рис.1.1. Создание сценария и его сохранение в рабочий каталог

3. В окне редактора прописала листинг по построению графика функции $y = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$ на интервале $[-10; 10]$ (рис.1.2).

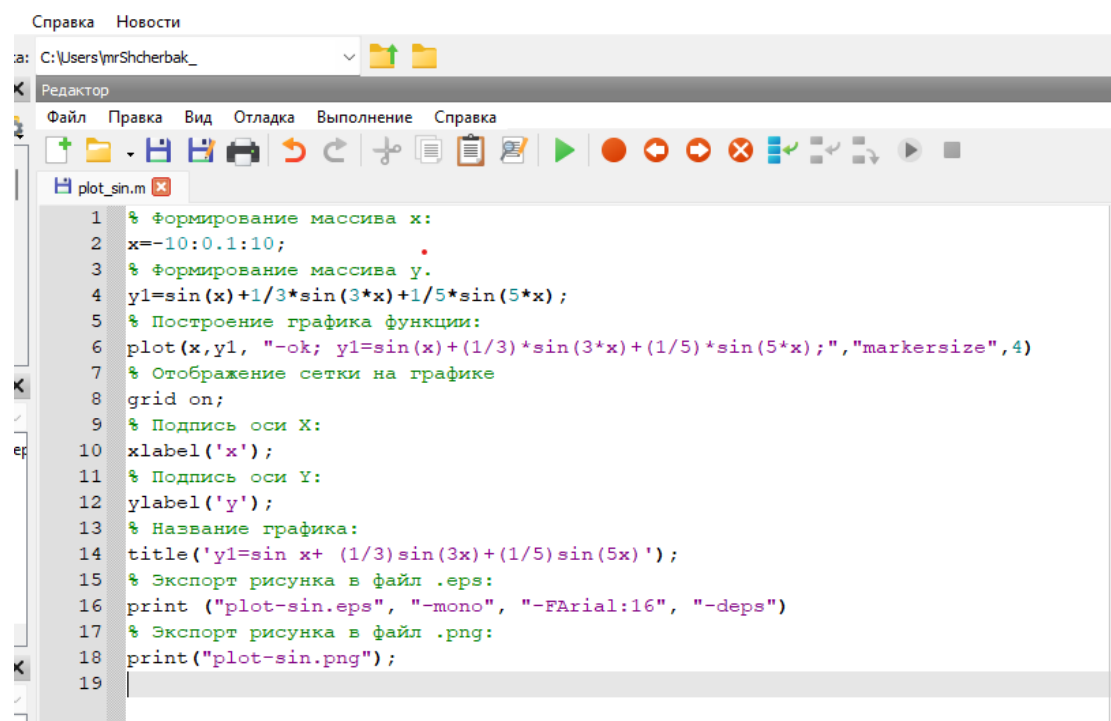


Рис.1.2. Листинг графика функции $y = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$

4. Запустила сценарий на выполнение. Открылось окно с построенным графиком (рис.1.3) и в моем рабочем каталоге появились файлы с графиками в форматах .eps,

.png (рис.1.4).

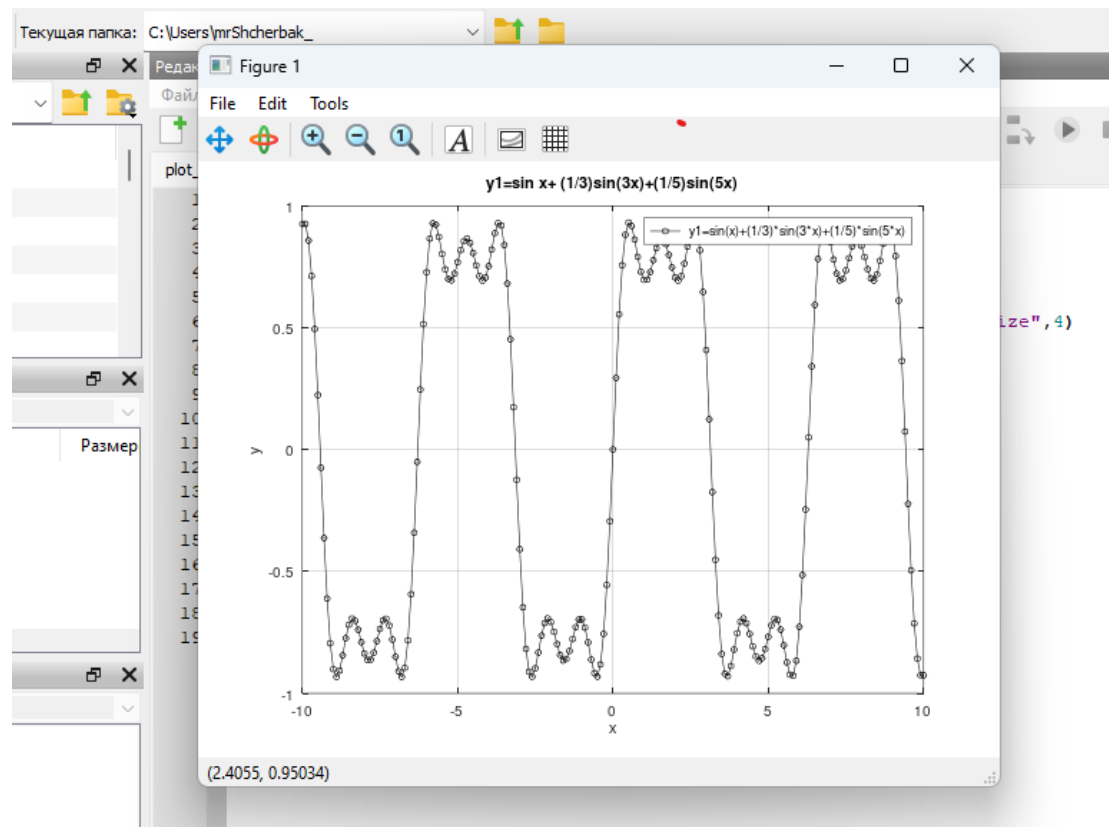


Рис.1.3. График функции $y = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$ на интервале $[-10; 10]$

```

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);", "r"
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", "-mono", "-FArial:16", "-deps");
  
```

Рис.1.4. Листинг графика функции $y = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$ и файлы с графиками в форматах .eps, .png

5. Создала сценарий под названием `plot_cos.m` и изменила его так, чтобы на одном графике располагались отличающиеся по типу линий графики функций $y_1 = \sin x + 1/3 \sin 3x + 1/5 \sin 5x$, $y_2 = \cos x + 1/3 \cos 3x + 1/5 \cos 5x$ (рис.1.5).

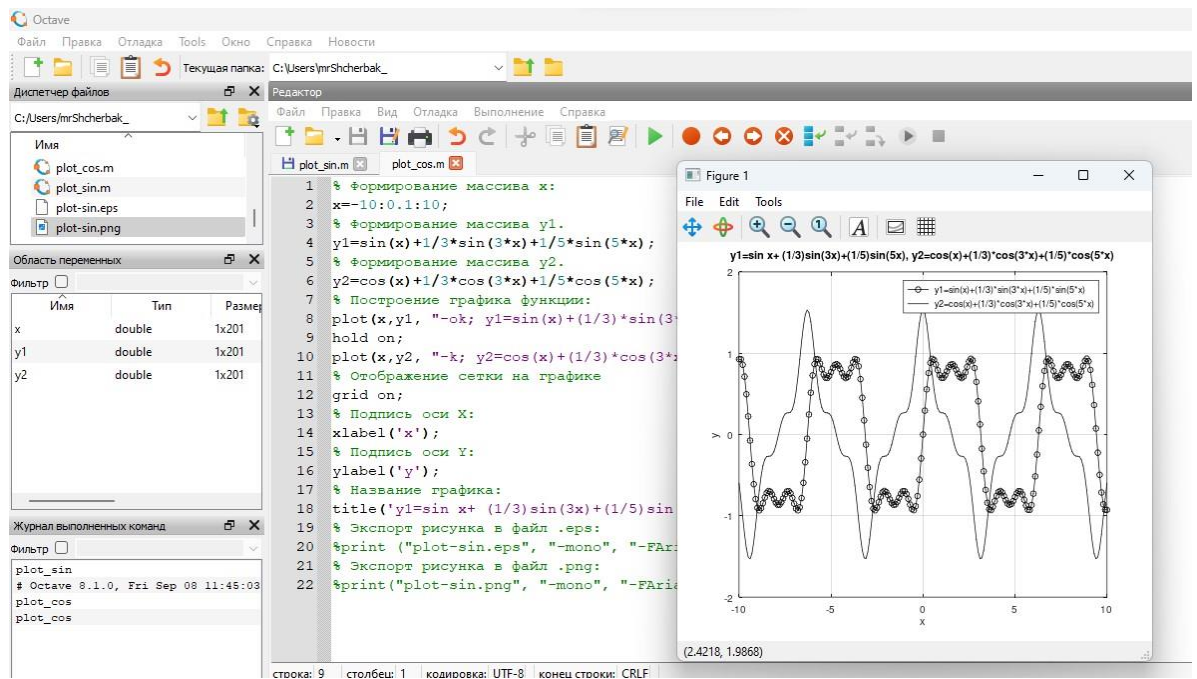


Рис.1.5. График функций y_1 и y_2 на интервале $[-10; 10]$

2. Разложение импульсного сигнала в частичный ряд Фурье

2.1. Постановка задачи

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

2.2. Выполнение

1. Создала новый сценарий с именем `meandr.m` и сохранила его в свой рабочий каталог.

2. В коде созданного сценария задала начальные значения:

`N=8;` % количество отсчетов (гармоник)

`t=-1:0.01:1;` % частота дискретизации

`A=1;` % значение амплитуды

`T=1;` % период

Гармоники, образующие меандр, имеют амплитуду, обратно пропорциональную номеру соответствующей гармоники в спектре. Для построения в одном окне отдельных графиков меандра с различным количеством гармоник реализовала суммирование ряда с накоплением и воспользовалась функциями `subplot` и `plot` для построения графиков. Продолжение листинга и результирующий график показаны на рис.2.1.

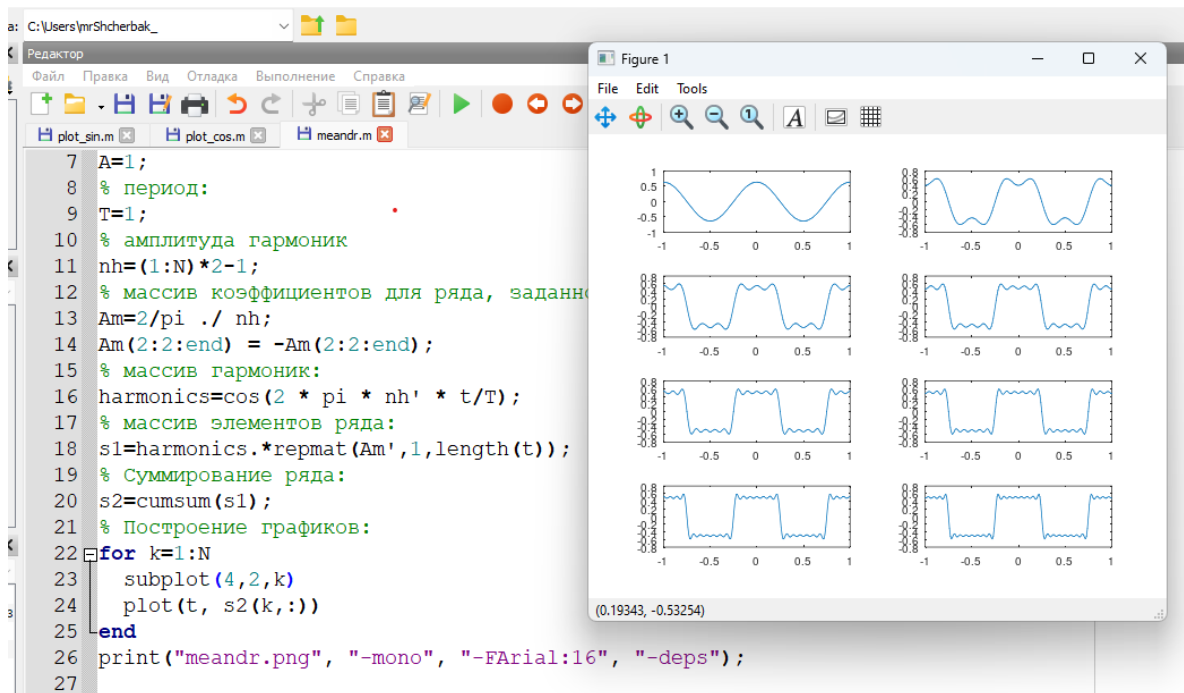


Рис.2.1. Листинг m-файла и графики меандра, реализованные с различным количеством гармоник

3. Экспортировала полученный график в файл в формате .png (рис.2.2).

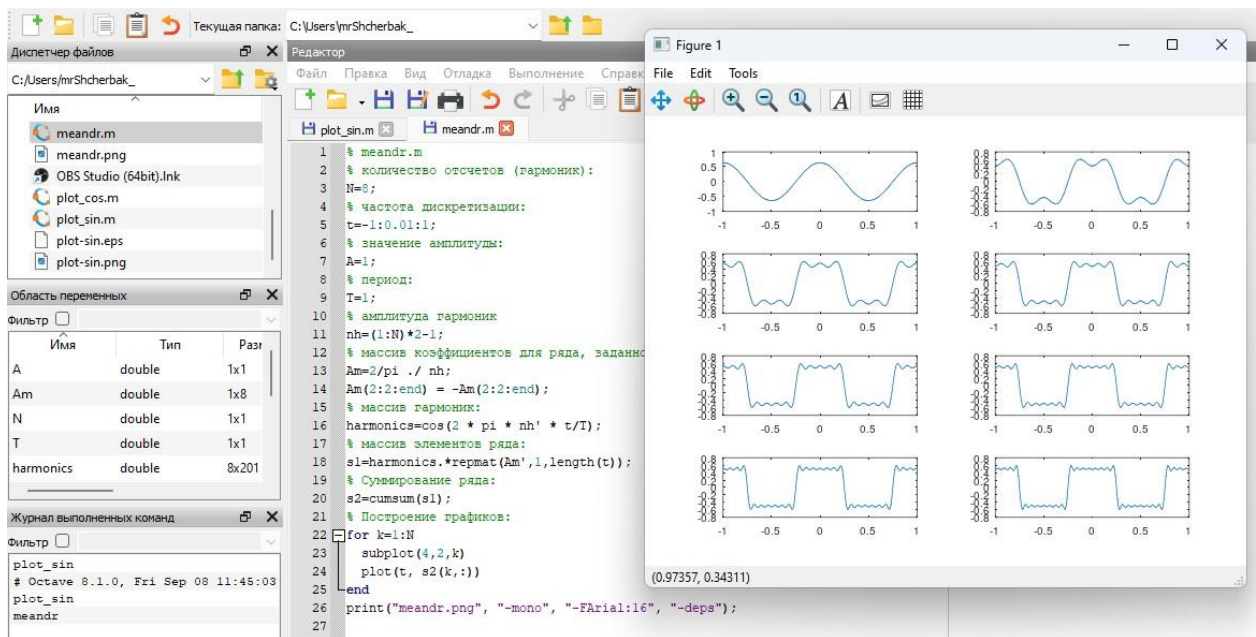


Рис.2.2. Листинг m-файла, графики меандра и файл meandr в формате .png

4.Скорректировала код для реализации меандра через синусы. Получила соответствующие графики (рис.2.3).

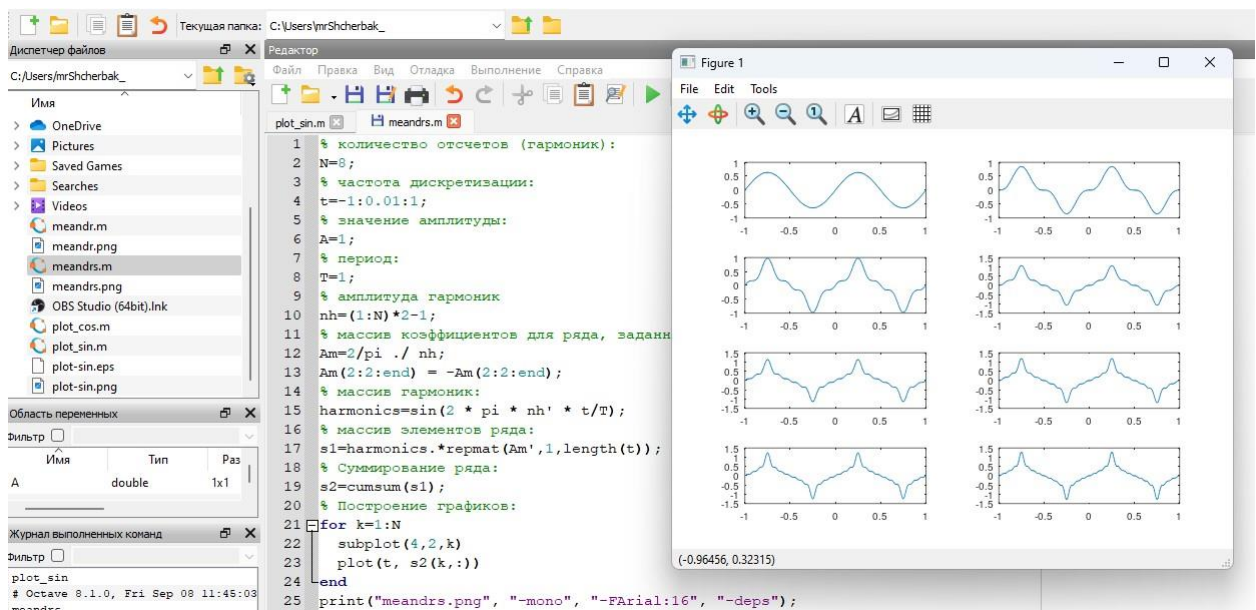


Рис.2.3. Листинг m-файла через синусы, графики меандра и файл meandrs в формате .png

3. Определение спектра и параметров сигнала

3.1. 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?

3.2. Выполнение

1. В своем рабочем каталоге создала каталог spectre1 и в нём новый сценарий с именем spectre.m.

2. В коде созданного сценария задала начальные значения, два синусоидальных сигнала разной частоты и построила графики сигналов (рис.3.1):

```
tmax = 0.5; % Длина сигнала (с)
```

```
fd = 512; % Частота дискретизации (Гц) (количество отсчётов)
```

```
f1 = 10; % Частота первого сигнала (Гц)
```

```
f2 = 40; % Частота второго сигнала (Гц):
```

```
a1 = 1; % Амплитуда первого сигнала
```

```
a2 = 0.7; % Амплитуда второго сигнала:
```

```
t = 0:1./fd:tmax; % Массив отсчётов времени
```

```
fd2 = fd/2; % Спектр сигнала
```

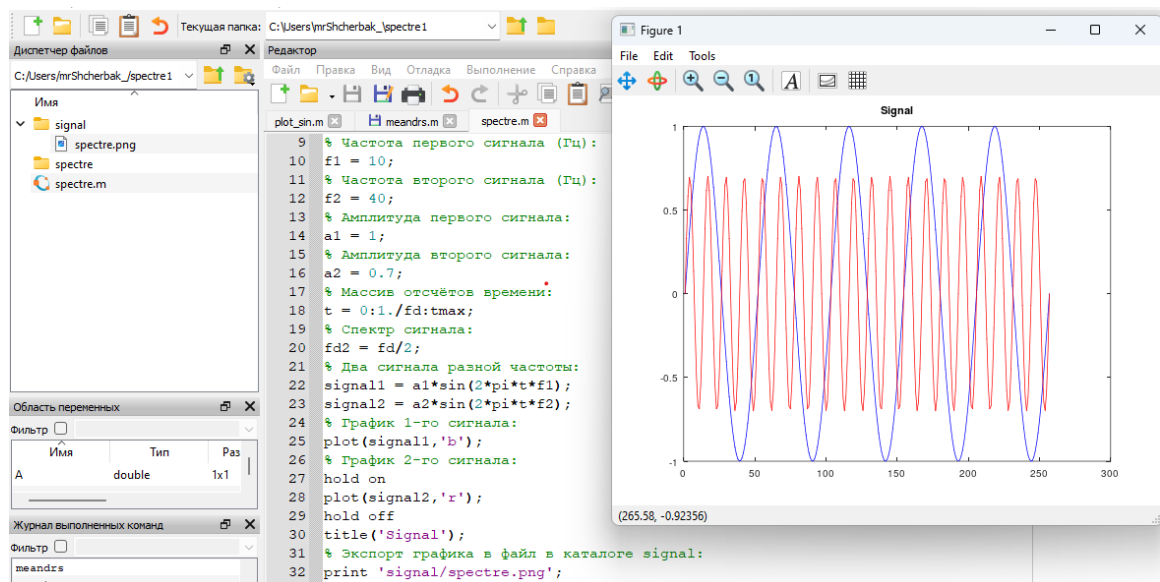


Рис.3.1. Листинг m-файла и два синусоидальных сигнала разной частоты

3. С помощью быстрого преобразования Фурье нашла спектры сигналов (рис. 3.2), добавив в файл spectre.m следующий код:

```
spectre1 = abs(fft(signal1,fd)); % Амплитуды преобразования Фурье сигнала 1
```



```
spectre2 = abs(fft(signal2,fd)); % Амплитуды преобразования Фурье сигнала 2
% Построение графиков спектров сигналов:
plot(spectre1,'b');
hold on plot(spectre2,'r');
hold off
title('Spectre');
print 'spectre/spectre.png';
```

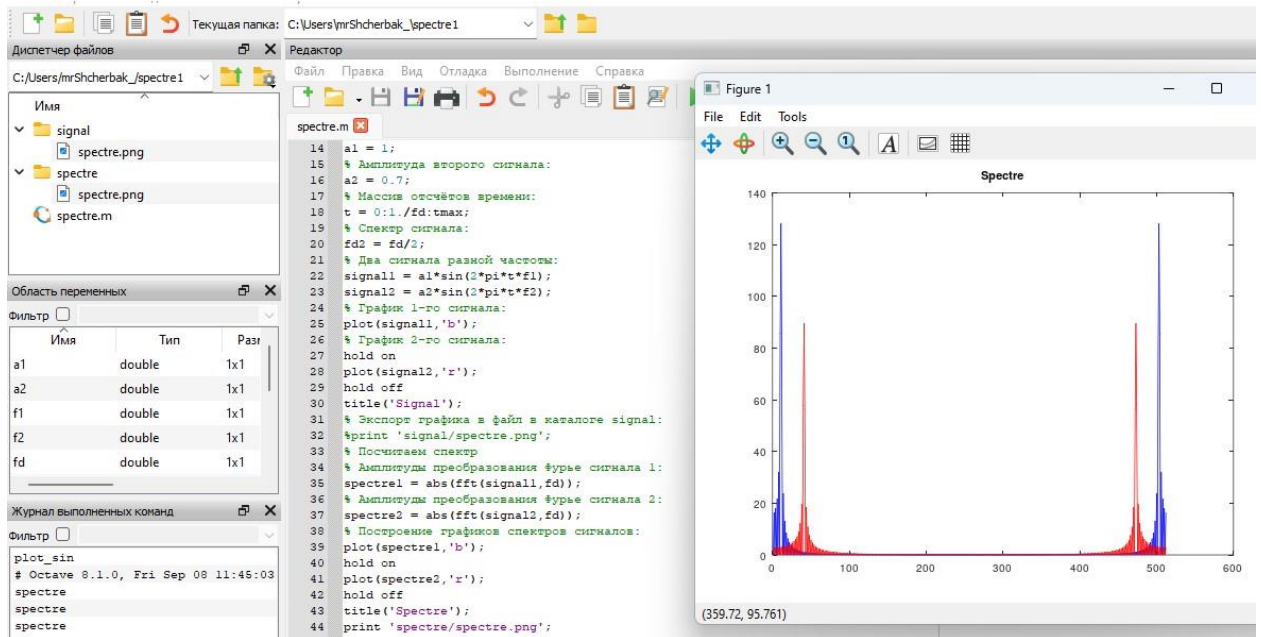


Рис.3.2. Листинг m-файла и график спектров синусоидальных сигналов

4.Скорректировала график спектра (рис. 3.3): отбросила дублирующие отрицательные частоты, а также добавила в файл spectre.m следующий код, учитывая, что на каждом шаге вычисления быстрого преобразования Фурье происходит суммирование амплитуд сигналов:

```
f = 1000*(0:fd2)./(2*fd); % Сетка частот
% 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';
```

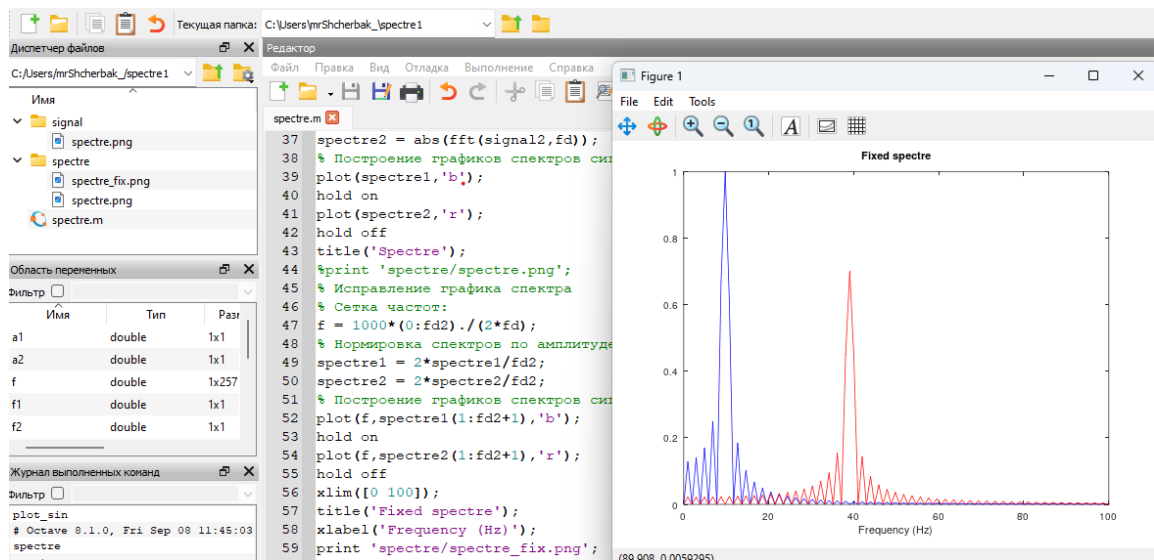


Рис.3.3. Листинг m-файла и исправленный график спектров синусоидальных сигналов

5. Нашла спектр суммы рассмотренных сигналов (рис. 3.4), создав каталог `spectr_sum` и файл в нём `spectre_sum.m` с кодом, представленным на рис.3.4.

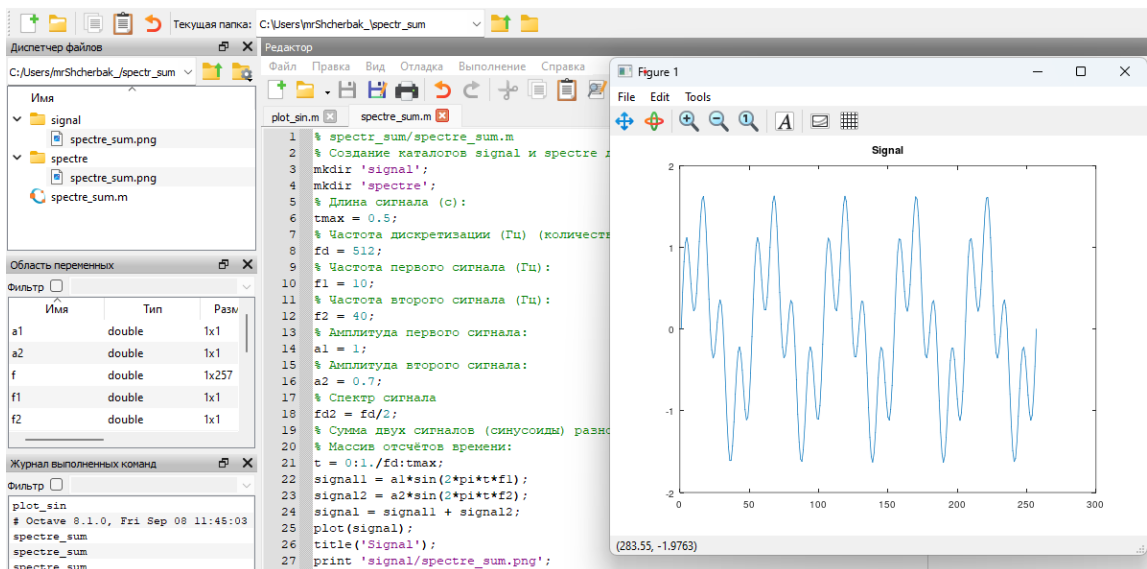


Рис.3.4. Листинг файла `spectre_sum.m` и суммарный сигнал

В результате получился аналогичный предыдущему результат (рис. 3.5), т.е. спектр суммы сигналов стал равен сумме спектров сигналов, что вытекает из свойств преобразования Фурье.

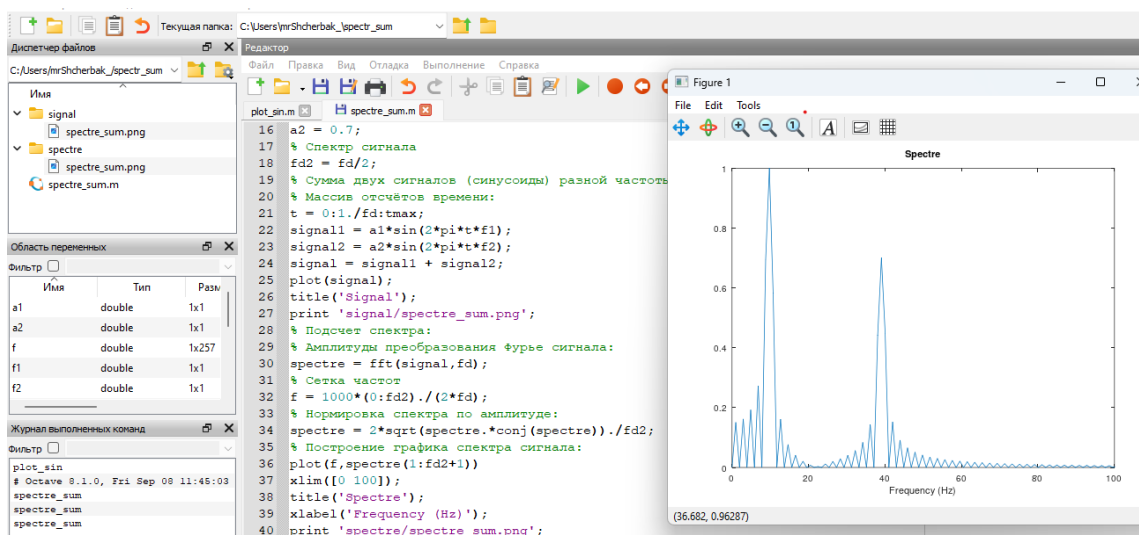


Рис.3.5. Листинг файла spectre_sum.m и спектр суммарного сигнала

Выполнила задание с другой частотой дискретизации – взяла значения 75 Гц и 1024 Гц (рис.3.6 – рис.3.7).

Если взять частоту дискретизации очень низкой, то произойдет так, что сигналы выше половины частоты дискретизации исказятся и будут неправильно интерпретироваться как низкочастотные сигналы. Это может привести к неверным результатам анализа спектра. Чтобы избежать этого, частота дискретизации должна быть выбрана достаточно высокой, чтобы учесть максимальную частоту в сигнале.

Если увеличивать частоту сигнала, то график этого сигнала будет более часто колебаться, что делает его более "зажатым" или "плотным" (рис.3.7). То есть будет больше колебаний за единицу времени, и это будет соответствовать более высокой частоте.

Если уменьшать частоту сигнала, график будет менее частым, с более широкими колебаниями за единицу времени (рис.3.6). Это будет соответствовать более низкой частоте.

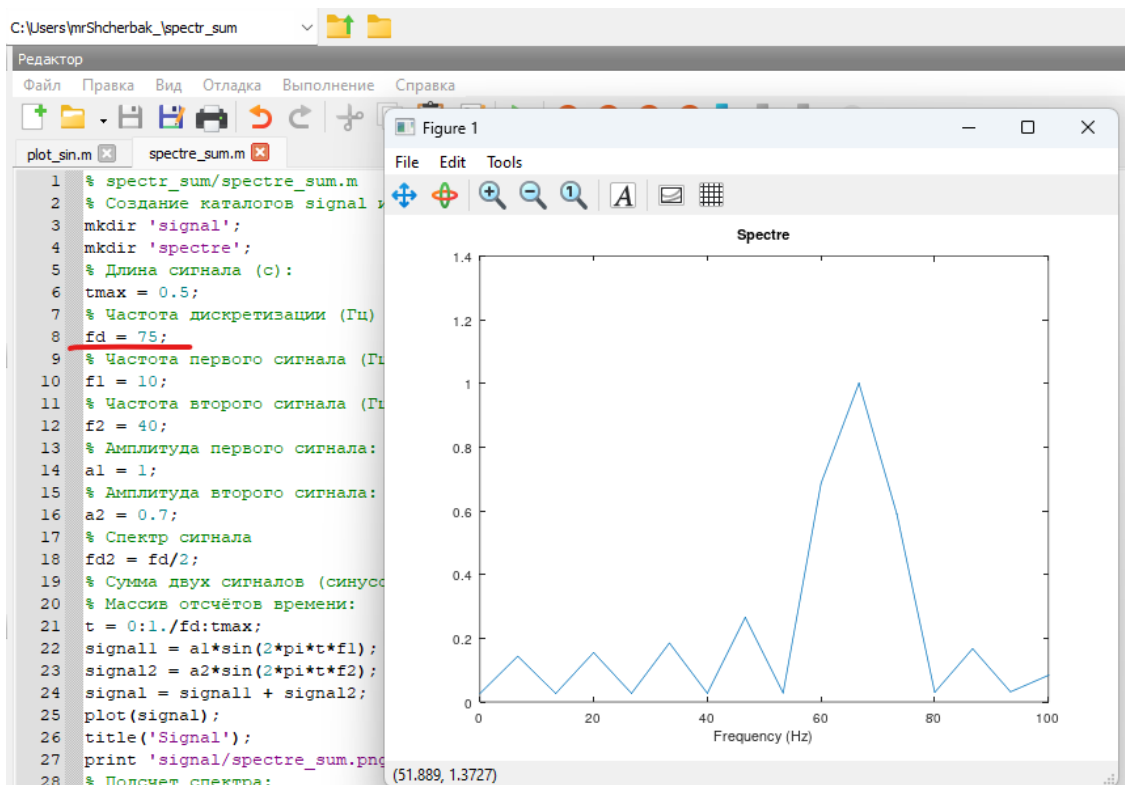


Рис.3.6. Листинг файла spectre_sum.m с частотой дискретизации $fd = 75$

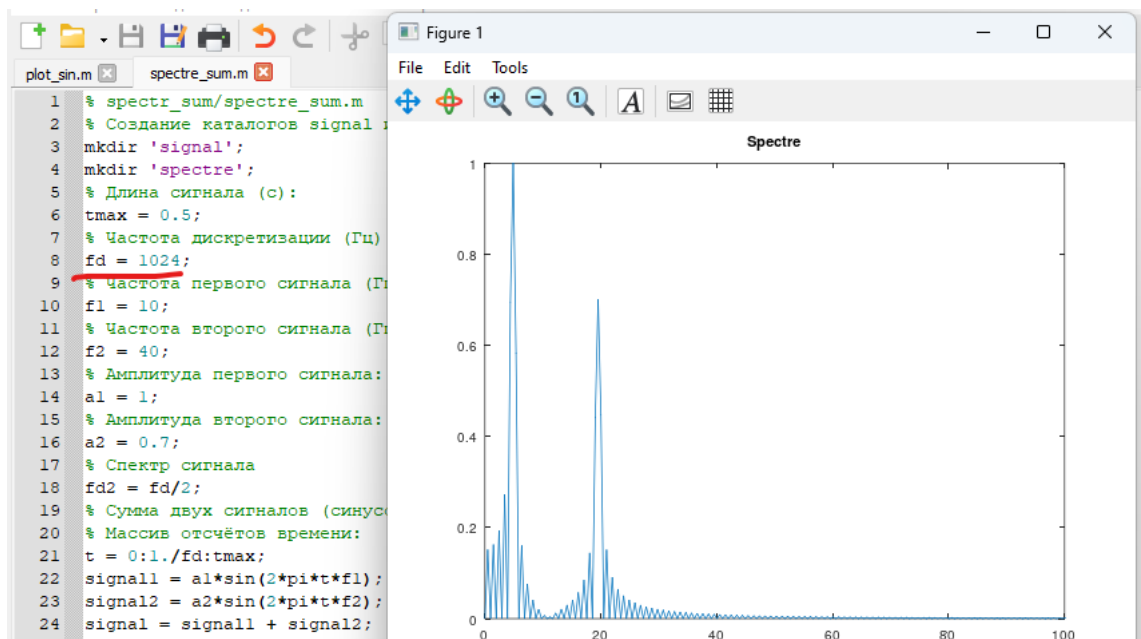


Рис.3.7. Листинг файла spectre_sum.m с частотой дискретизации $fd = 1024$

4. Амплитудная модуляция

4.1. Постановка задачи

Продемонстрировать принципы модуляции сигнала на примере аналоговой

амплитудной модуляции (рис.4.1).

4.2. Выполнение

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

2. Добавила в файле am.m следующий код:

```
% Создание каталогов signal и spectre для размещения графиков:

mkdir 'signal';

mkdir 'spectre';

% Модуляция синусоид с частотами 50 и 5

tmax = 0.5; % Длина сигнала (с)

fd = 512; % Частота дискретизации (Гц) (количество отсчётов)

f1 = 5; % Частота сигнала (Гц)

f2 = 50; % Частота несущей (Гц)

fd2 = fd/2; % Спектр сигнала

% Построение графиков двух сигналов (синусоиды) разной частоты

t = 0:1./fd:tmax; % Массив отсчётов времени:

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);
```

```
f = 1000*(0:fd2)./(2*fd); % Сетка частот
```

```
spectre = 2*sqrt(spectre.*conj(spectre))./fd2; % Нормировка спектра по амплитуде
```

% Построение спектра:

```
plot(f,spectre(1:fd2+1), 'b')
```

```
xlim([0 100]);
```

```
title('Spectre');
```

```
xlabel('Frequency (Hz)');
```

```
print 'spectre/am.png';
```

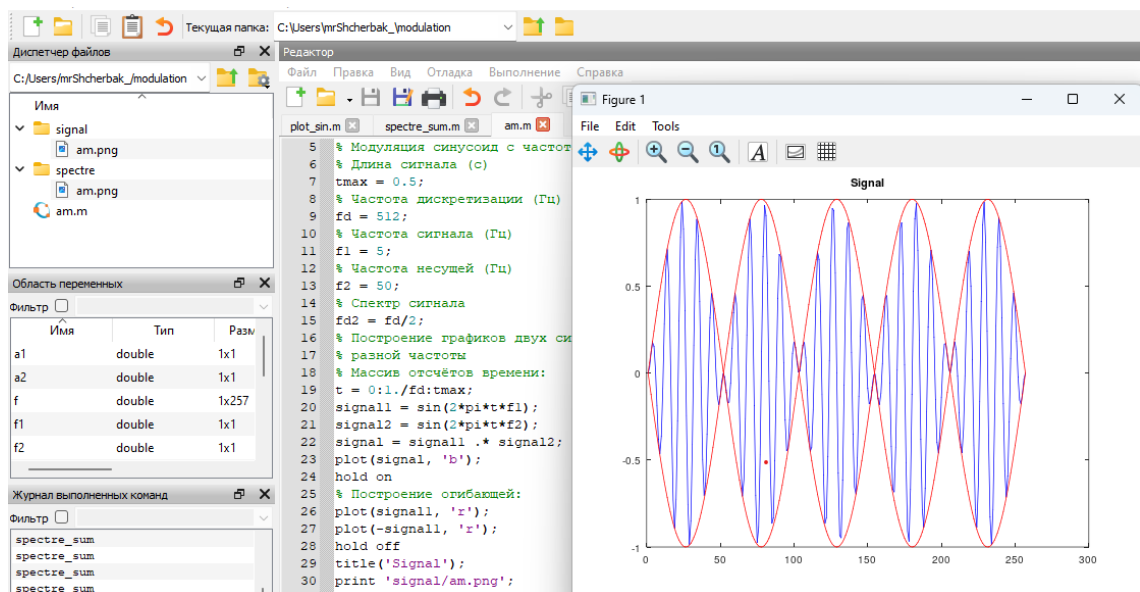


Рис.4.1. Листинг файла am.m. Сигнал и огибающая при амплитудной модуляции

В результате получила, что спектр произведения представляет собой свёртку спектров (рис.4.2).

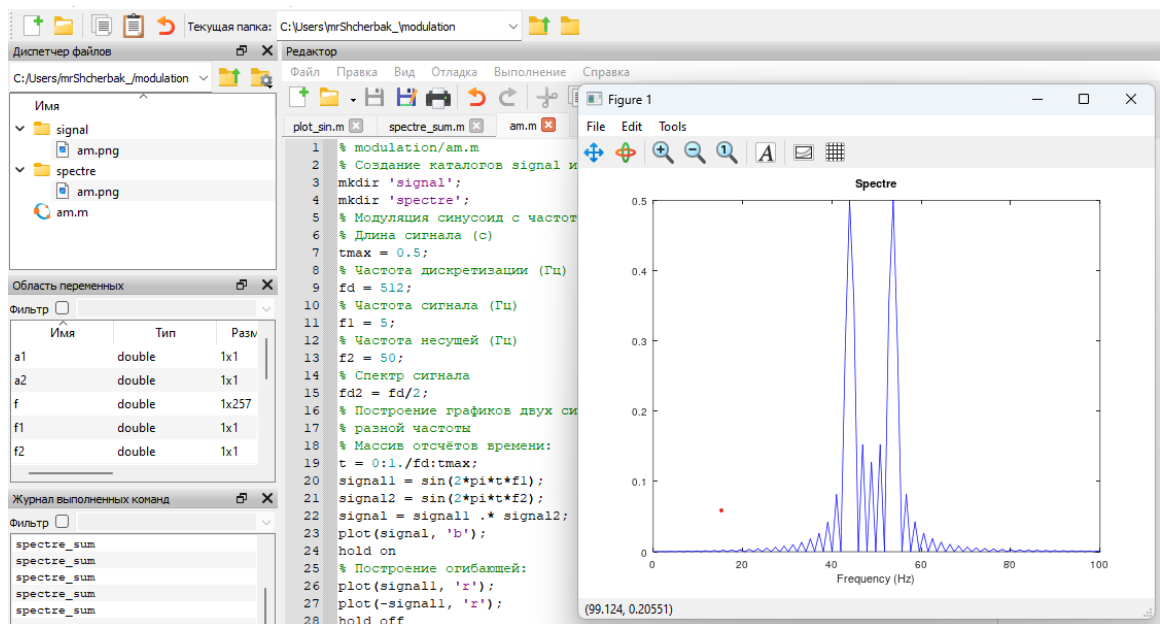


Рис.4.2. Листинг файла am.m и спектр сигнала при амплитудной модуляции

5. Кодирование сигнала. Исследование свойства самосинхронизации сигнала

5.1. Постановка задачи

Требуется получить кодированные сигналы для нескольких кодов, проверить свойства самосинхронизуемости кодов, получить спектры.

5.2. Выполнение

1. В своем рабочем каталоге создала каталог coding и в нём файлы main.m, maptowave.m, unipolar.m, ami.m, bipolarnrz.m, bipolarrrz.m, manchester.m, diffmanc.m, calcspectre.m (рис.5.1).

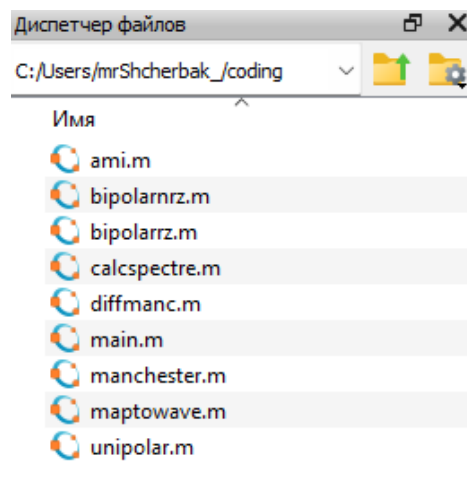


Рис.5.1. Необходимые файлы в каталоге coding

2. В окне интерпретатора команд проверила, установлен ли у меня пакет расширений signal (рис.5.2).

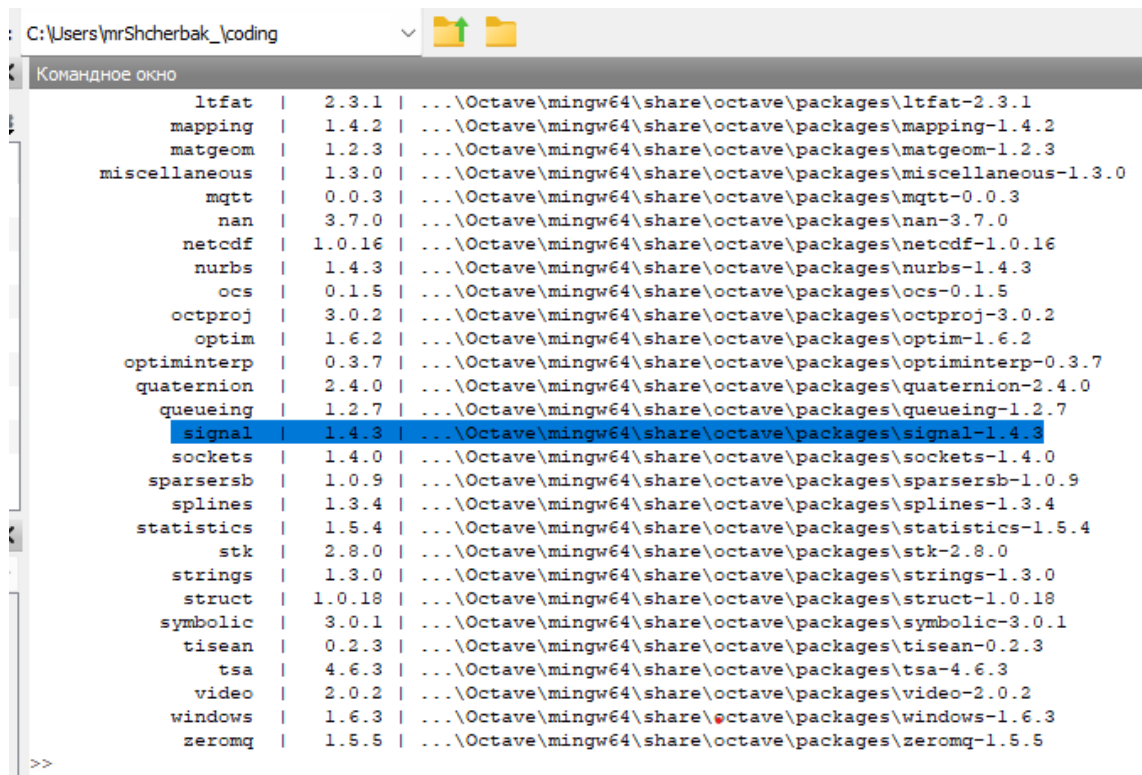


Рис.5.2. Установлен пакет расширений signal

3. В файле main.m подключила пакет signal и задала входные кодовые последовательности: 1-14 строки листинга рис. 5.3.

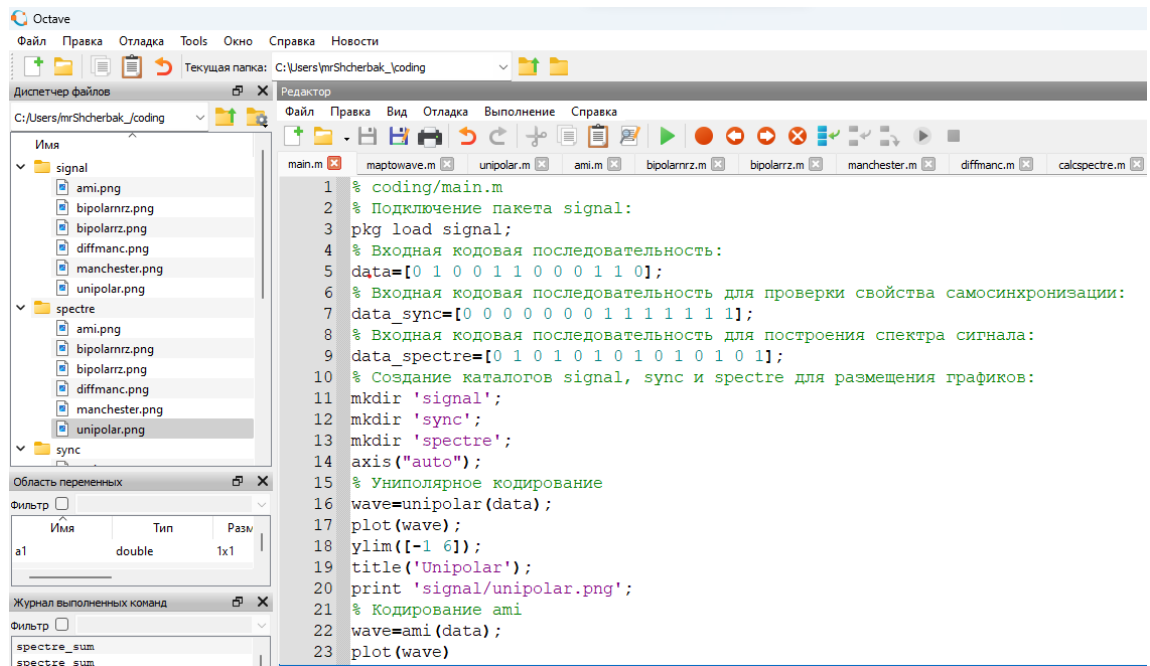


Рис.5.3. Листинг файла main.m

Затем в этом же файле прописала вызовы функций для построения графиков модуляций кодированных сигналов для кодовой последовательности 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';
```

Затем в этом же файле прописала вызовы функций для построения графиков модуляций кодированных сигналов для кодовой последовательности data_sync:

% 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';
Далее в этом же файле прописала вызовы функций для построения графиков
спектров:
% 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');
```

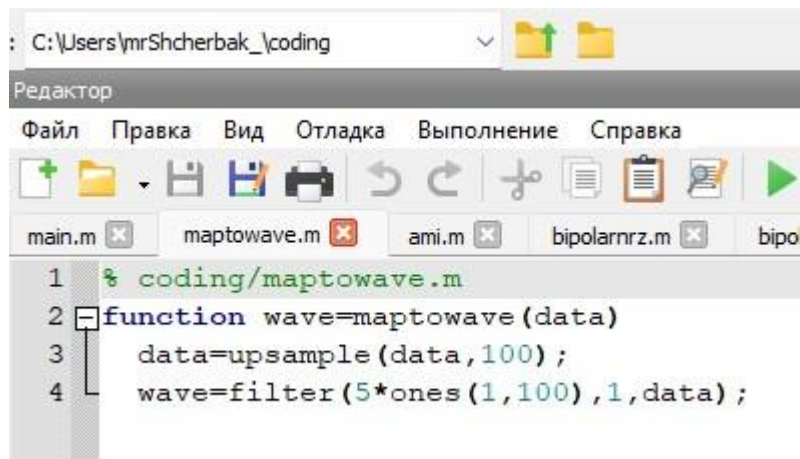
```
print 'spectre/manchester.png';
```

% Дифференциальное манчестерское кодирование:

```
wave=diffmanc(data_spectre);  
spectre=calcspectre(wave);  
title('Differential Manchester');
```

```
print 'spectre/diffmanc.png';
```

4. В файле `maptowave.m` прописала функцию, которая по входному битовому потоку строит график сигнала (рис.5.4).

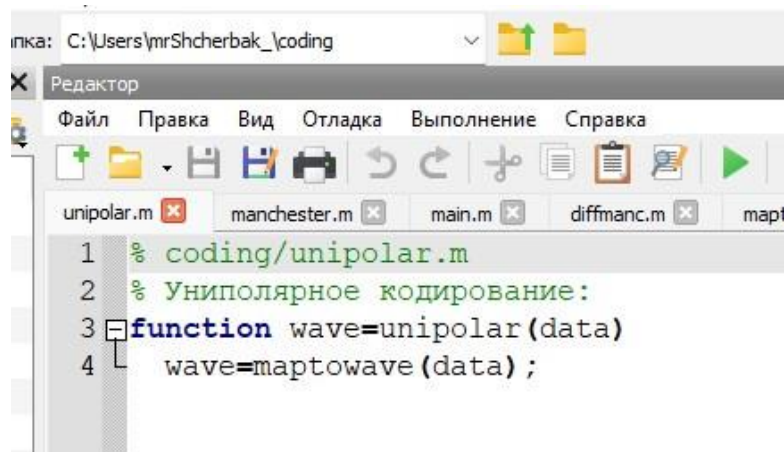


The screenshot shows the MATLAB editor window with the file path C:\Users\mrShcherbak_\coding. The editor displays the content of the file maptowave.m. The code defines a function wave=maptowave(data) that upsamples the data by 100 and then filters it with a 5th-order Butterworth low-pass filter.

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

Рис.5.4. Содержимое файла maptowave.m

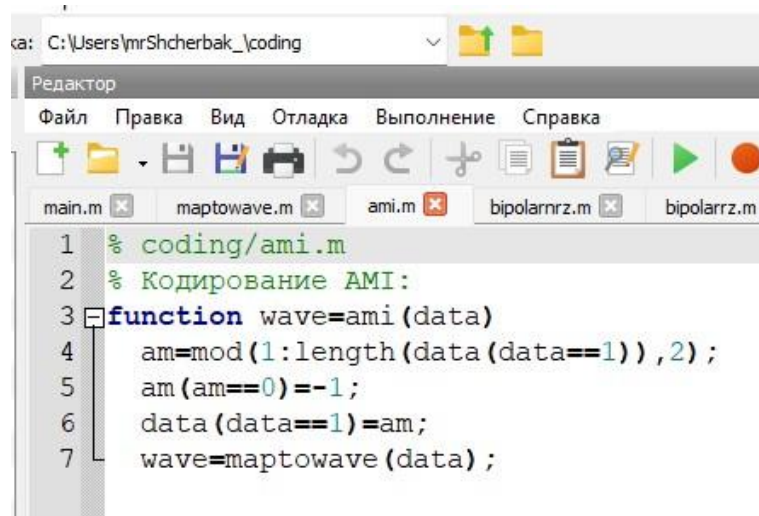
5. В файлах unipolar.m, ami.m, bipolarnrz.m, bipolarrrz.m, manchester.m, diffmanc.m прописала соответствующие функции преобразования кодовой последовательности data с вызовом функции maptowave для построения соответствующего графика (рис.5.6 – рис.5.10).



The screenshot shows the MATLAB editor window with the file path C:\Users\mrShcherbak_\coding. The editor displays the content of the file unipolar.m. The code defines a function wave=unipolar(data) that calls the maptowave function to generate the waveform.

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

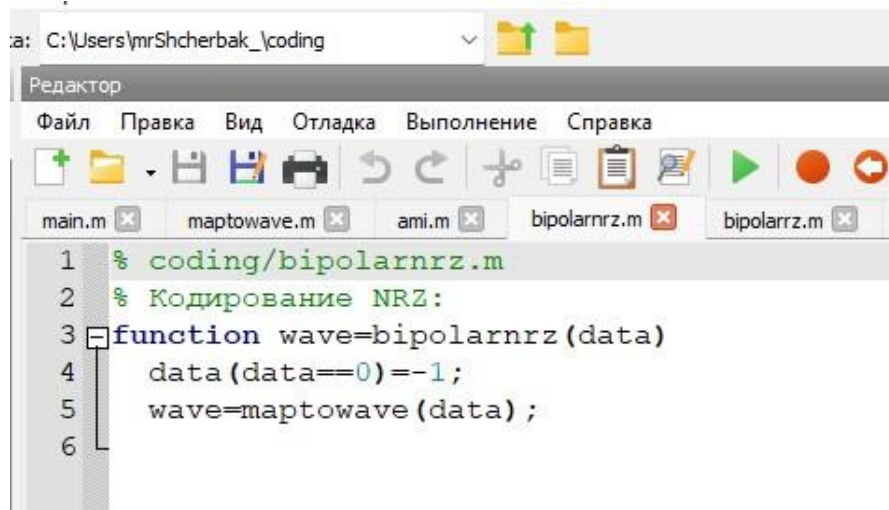
Рис.5.5. Содержимое файла unipolar.m



The screenshot shows the MATLAB editor window with the file path C:\Users\mrShcherbak_\coding. The editor displays the content of the file ami.m. The code defines a function wave=ami(data) that generates the AMI waveform by first creating a bipolar signal and then calling the maptowave function.

```
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);
```

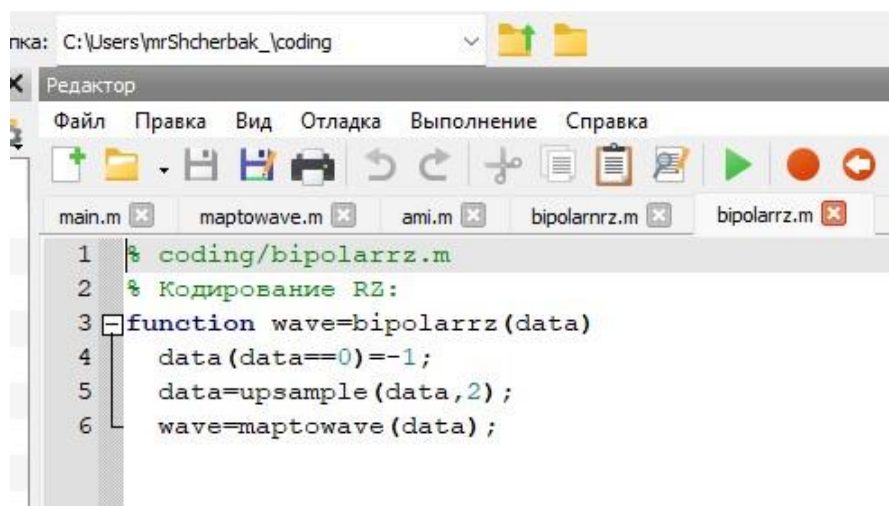
Рис.5.6. Содержимое файла ami.m



The screenshot shows the MATLAB editor window with the file path 'C:\Users\mrShcherbak_\coding'. The editor displays the code for the function 'bipolarnrz.m'. The code is as follows:

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

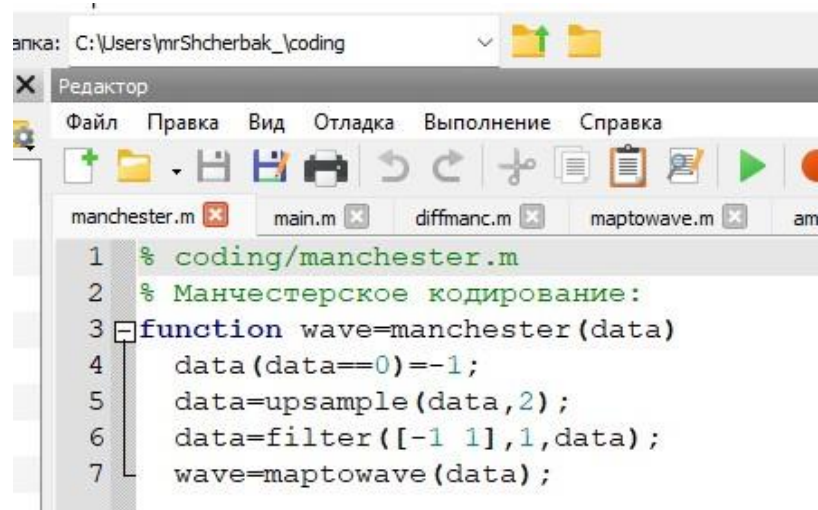
Рис.5.7. Содержимое файла bipolarnrz.m



The screenshot shows the MATLAB editor window with the file path 'C:\Users\mrShcherbak_\coding'. The editor displays the code for the function 'bipolarrrz.m'. The code is as follows:

```
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);
```

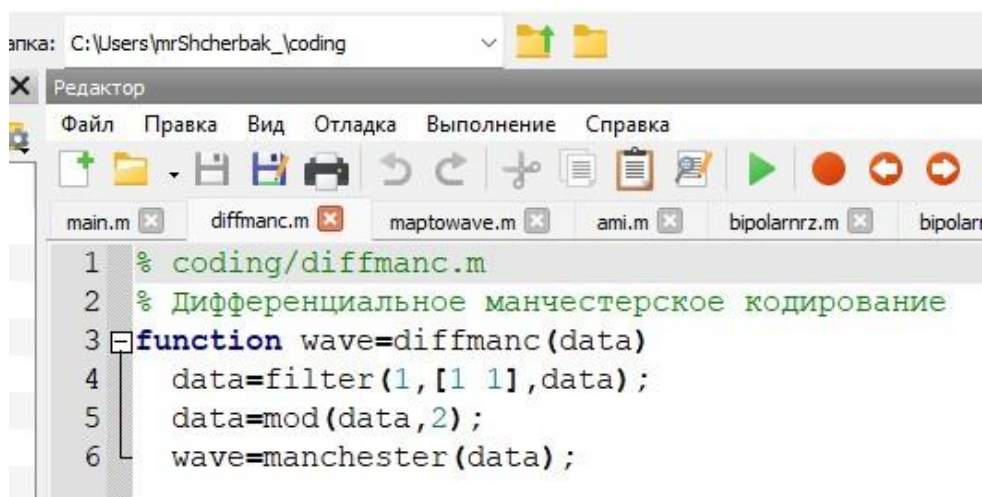
Рис.5.8. Содержимое файла bipolarrrz.m



The screenshot shows the MATLAB editor window with the file path 'C:\Users\mrShcherbak_\coding'. The editor displays the code for the function 'manchester.m'. The code is as follows:

```
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);
```

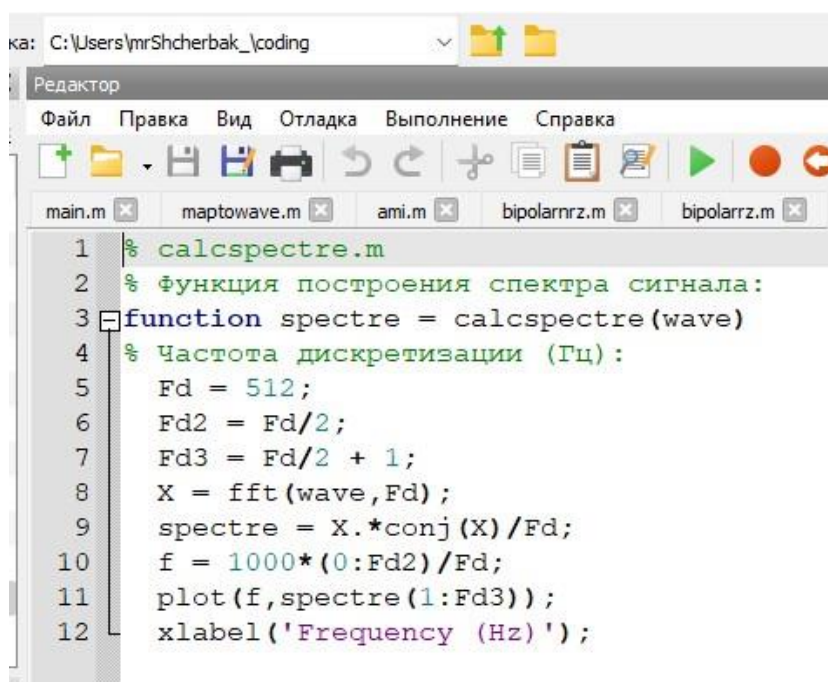
Рис.5.9. Содержимое файла manchester.m



```
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);
```

Рис.5.10. Содержимое файла diffmanc.m

6. В файле calcspectre.m прописала функцию построения спектра сигнала (рис.5.11).



```
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)');
```

Рис.5.11. Содержимое файла calcspectre.m

7. Запустила главный скрипт main.m. В каталоге signal получены файлы с графиками кодированного сигнала (, в каталоге sync — файлы с графиками, иллюстрирующими свойства самосинхронизации, в каталоге spectre — файлы с графиками спектров сигналов. Некоторые из графиков показаны на рис. 5.12 – рис.5.15.

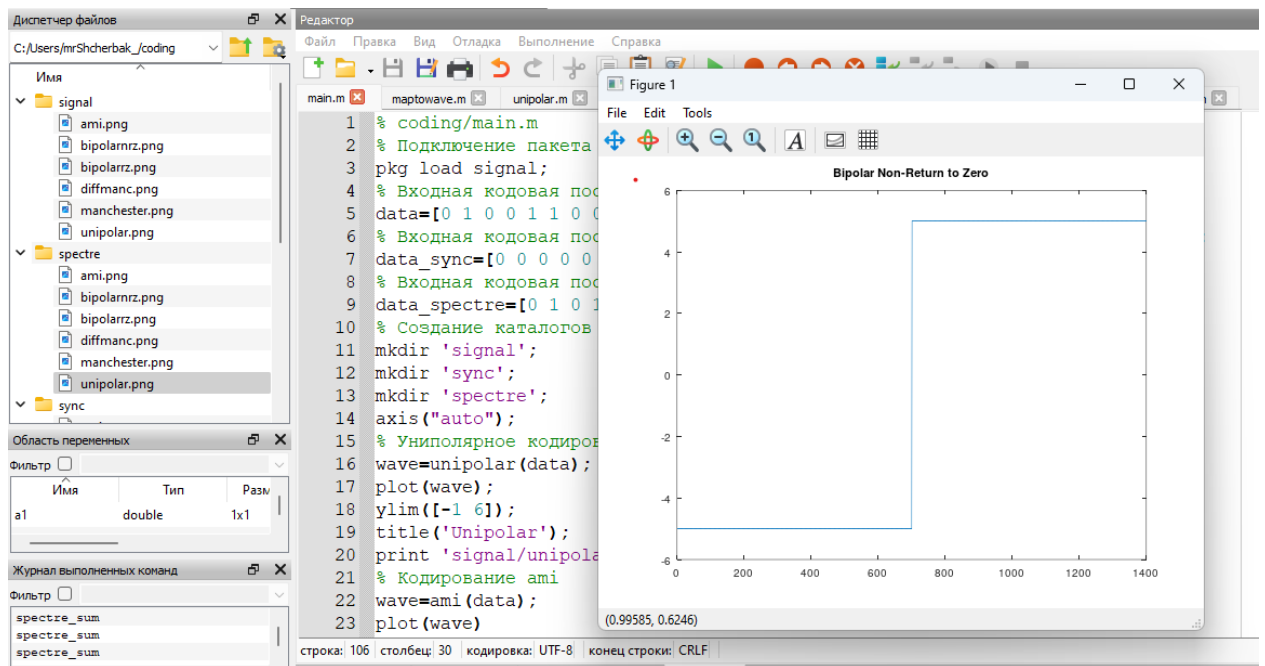


Рис.5.12. Кодирование NRZ: нет самосинхронизации

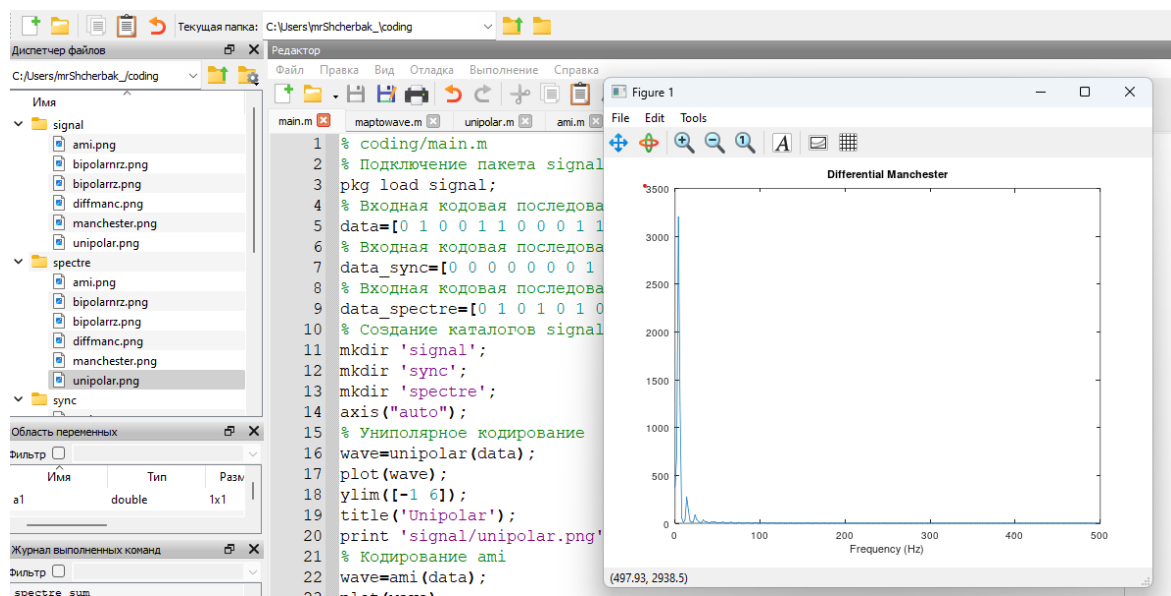


Рис.5.13. Дифференциальное манчестерское кодирование: спектр сигнала

модуляции сигнала на примере аналоговой амплитудной модуляции. Исследовала свойства самосинхронизации сигнала.