Лабораторная работа 1

Методы кодирования и модуляция сигналов

Лушин Артём Андреевич

Содержание

Список иллюстраций

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 лабораторной работы

1) Я создал и сохранил файл plot_sin.m. В этом файле написал код, чтобы построить функцию. После исполнения кода вывелся график.

```
x=-10:0.1:10;
y1=sin(x)+1/3*sin(3*x)+1/5*sin(5*x);
plot(x,y1, "-ok; y1=sin(x)+(1/3)*sin(3*x)+(1/5)*sin(5*x);","markersize",4)
grid on;
xlabel('x');
ylabel('y');
title('y1=sin x+ (1/3)sin(3x)+(1/5)sin(5x)');
print ("plot-sin.eps", "-mono", "-FArial:16", "-deps")
print("plot-sin.png");
```

Рис. 2.1: Код plot_sin.m

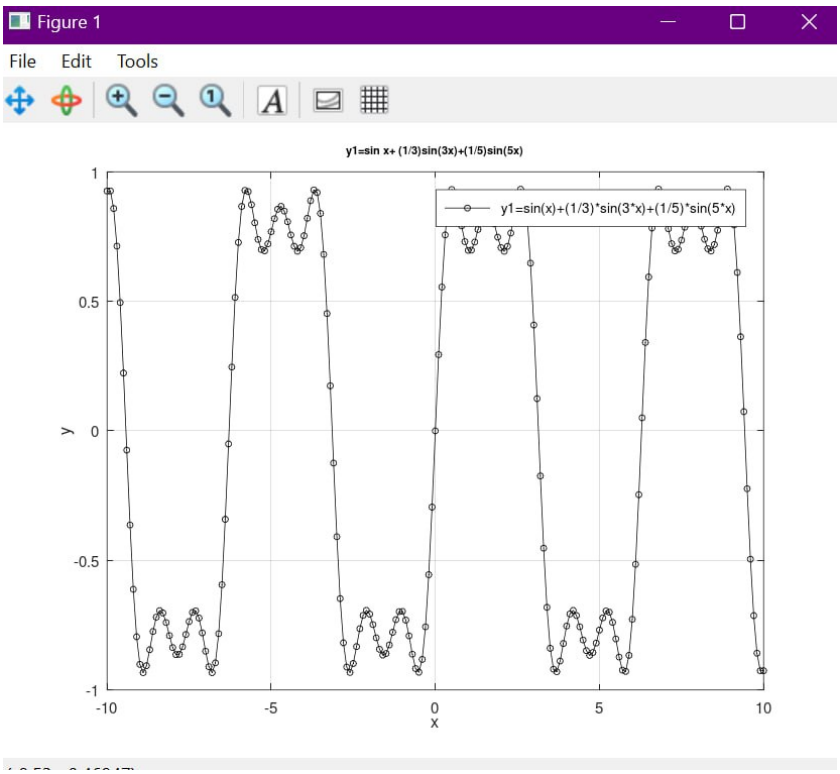


Рис. 2.2: График plot_sin.m

2) Создали новый файл и написал код, который основывается на коде из прошлого пункта, но добавляет и функцию с косинусом. Сгенерировал рисунок.

```
x=-10:0.1:10;
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);
plot(x,y1, "-ok; y1=sin(x)+(1/3)*sin(3*x)+(1/5)*sin(5*x);","markersize",4)
hold on;
plot(x,y2, "-k; y2=cos(x)+1/3*cos(3*x)+1/5*cos(5*x);","markersize",4)
grid on;
xlabel('x');
ylabel('y');
title('y1=sin x+ (1/3)sin(3x)+(1/5)sin(5x)');
title('y2=cos(x)+1/3*cos(3*x)+1/5*cos(5*x)');
print ("plot-sin.eps", "-mono", "-FArial:16", "-deps")
print("plot-sin.png");
```

Рис. 2.3: Код plot_combo.m

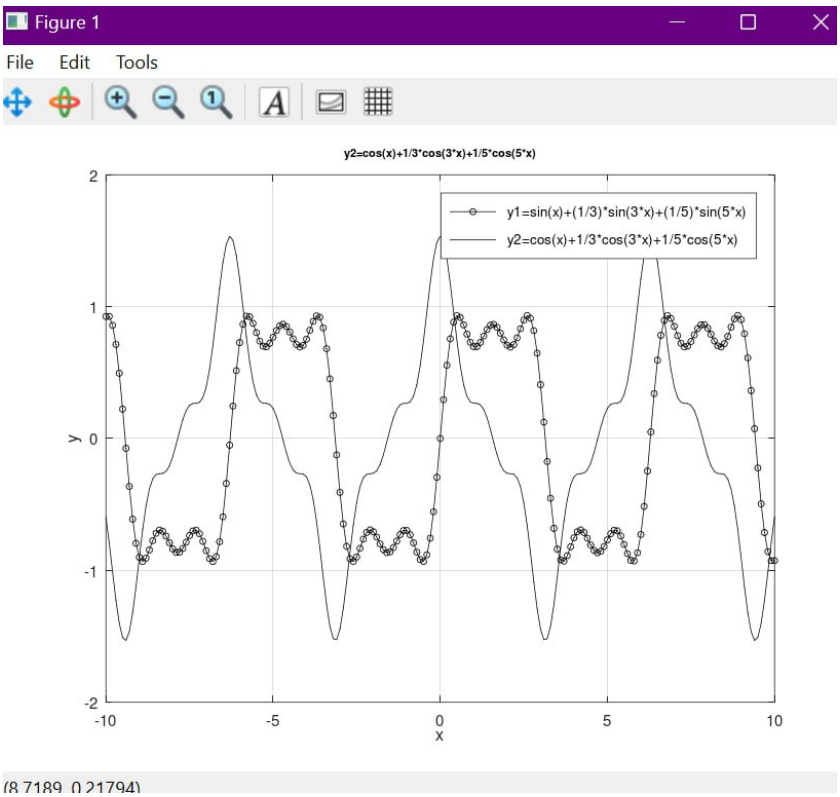


Рис. 2.4: Рисунок plot_combo.m

3) Создал новый файл meandr для реализации графики меандра, реализованный с различным количеством гармоник. Реализовывал через косинусы.

```
% 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,:))
end
```

Рис. 2.5: Код meandr.m

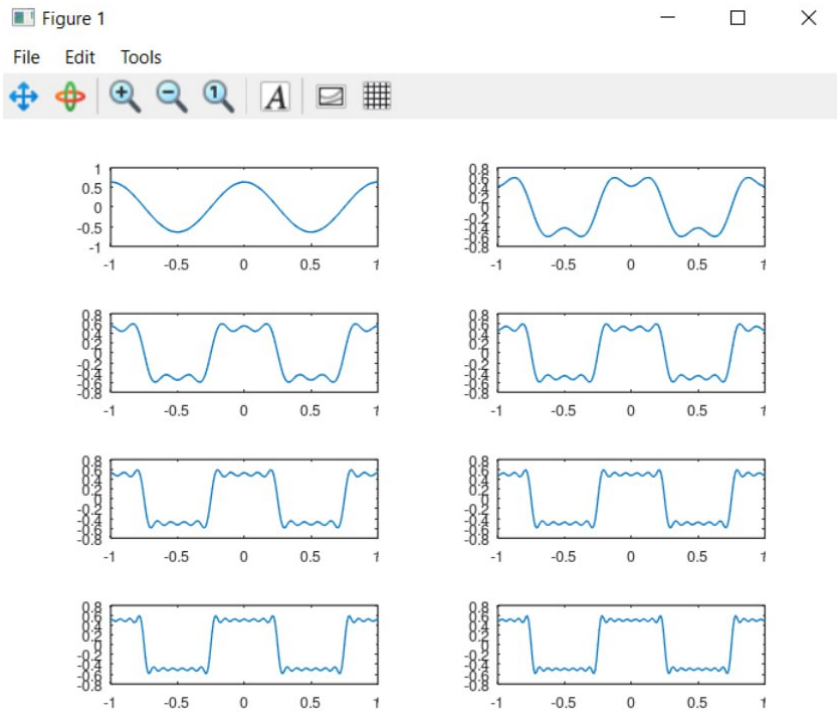


Рис. 2.6: Рисунок meandr.m

4) Изменил код файла meandr, чтобы он реализовывался через синусы.

```
% 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,:))
end
```

Рис. 2.7: Код изменённого meandr.m

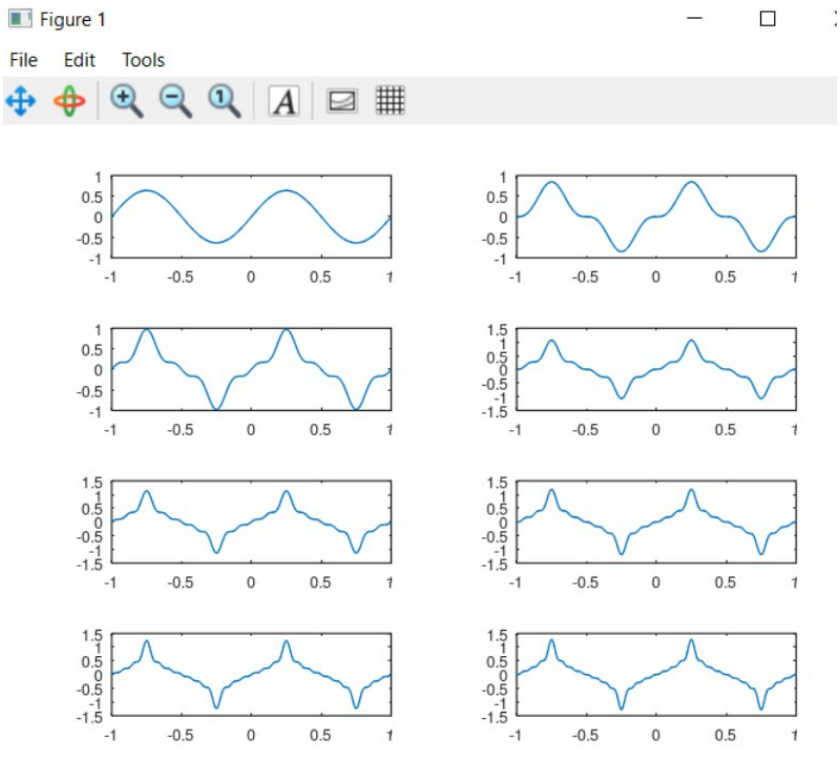


Рис. 2.8: Рисунок изменённого meandr.m

5) Создал новый каталог spectre1, а в нём файл spectre.m. Файл будет определять спектр двух отдельных сигналов и их сумму.

```
% 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';

```

Рис. 2.9: Код spectre.m

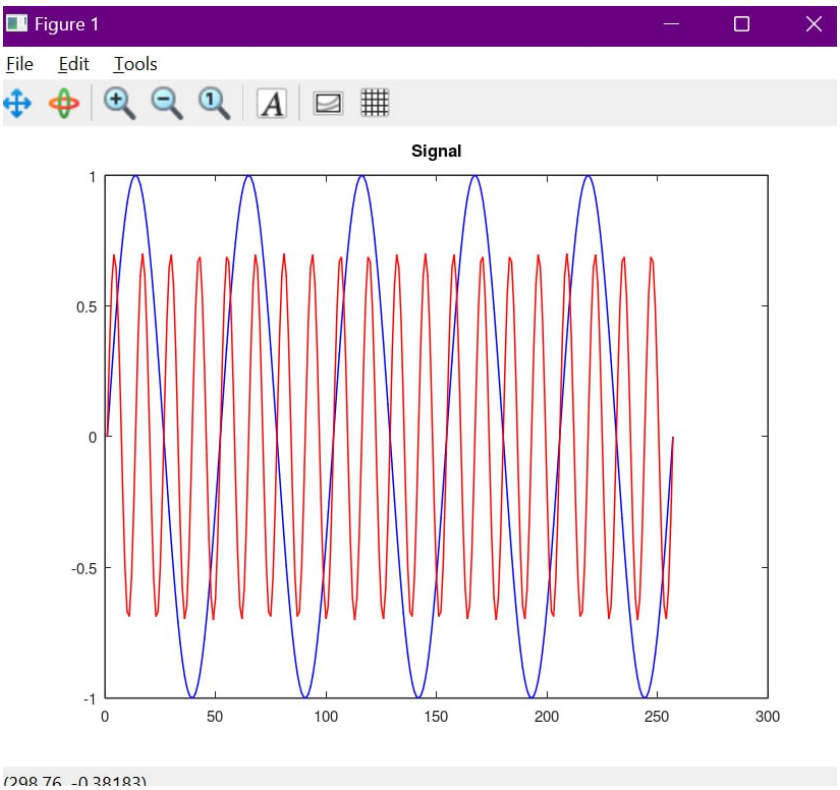


Рис. 2.10: Рисунок spectre.m

6) Затем изменил код файла spectre, чтобы он использовал преобразования Фурье.

```
% 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';
```

Рис. 2.11: Код spectre.m с Фурье

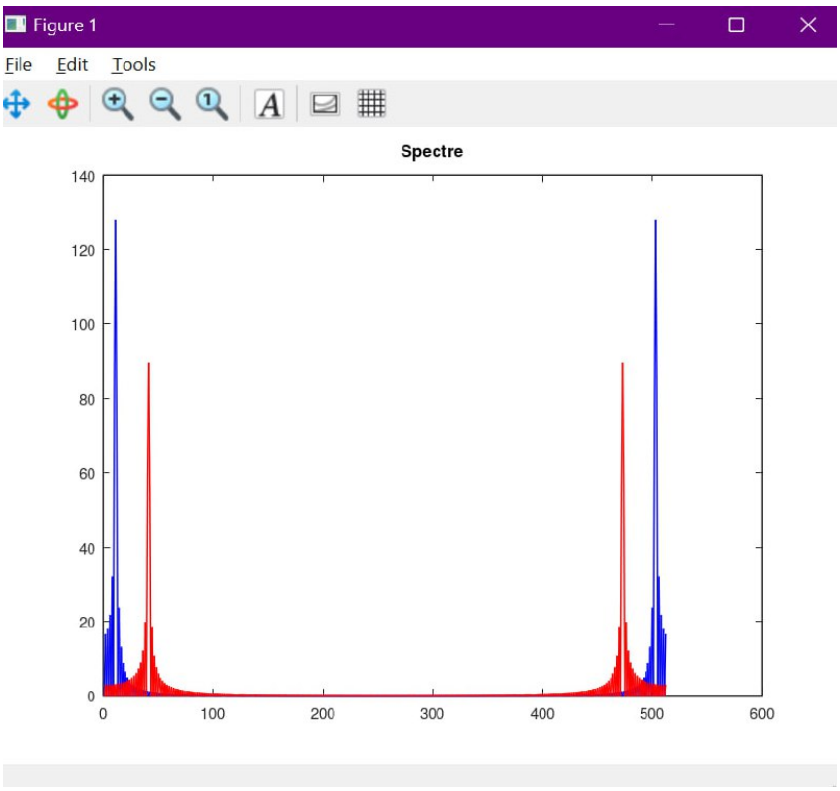


Рис. 2.12: Рисунок spectre.m с Фурье

7) Я улучшил график преобразования Фурье, чтоб он отбрасывал дублирующие отрицательные частоты.

```
% 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. 2.13: Код spectre.m с улучшенным Фурье

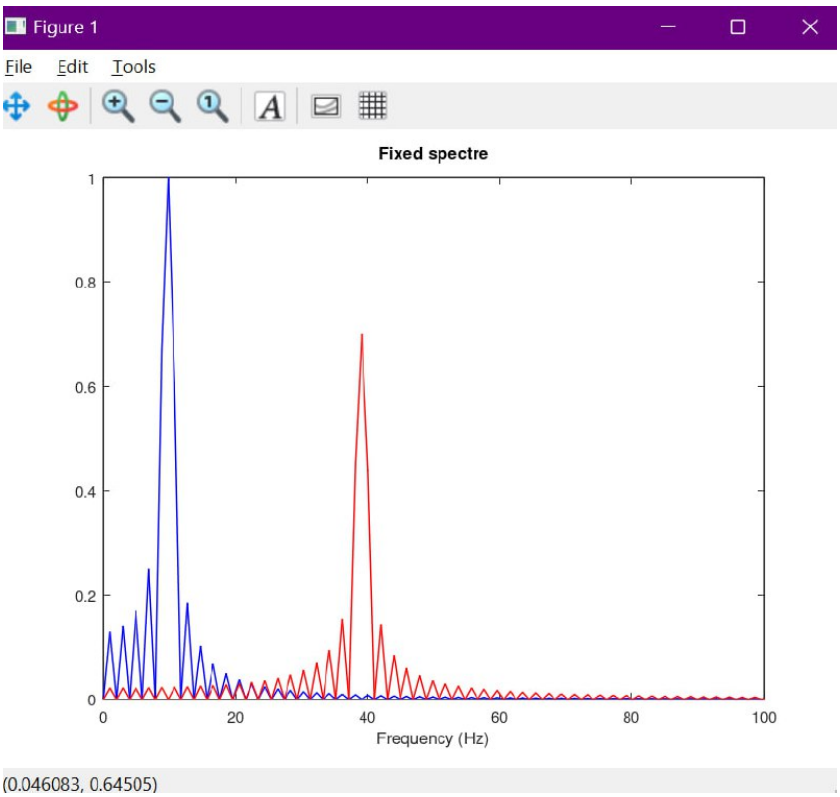


Рис. 2.14: Рисунок spectre.m с улучшенным Фурье

8) Я создал каталог spectr_sum и в нём создал файл spectre_sum.m. Вписал следующий код и на выходе доказал, что спектр сумм сигналов равен сумме спектров сигнала.

```
% spectr_sum/spectre_sum.m
% Создание каталогов signal и spectre для размещения dграфиков:
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';
```

Рис. 2.15: Код spectre_sum.m

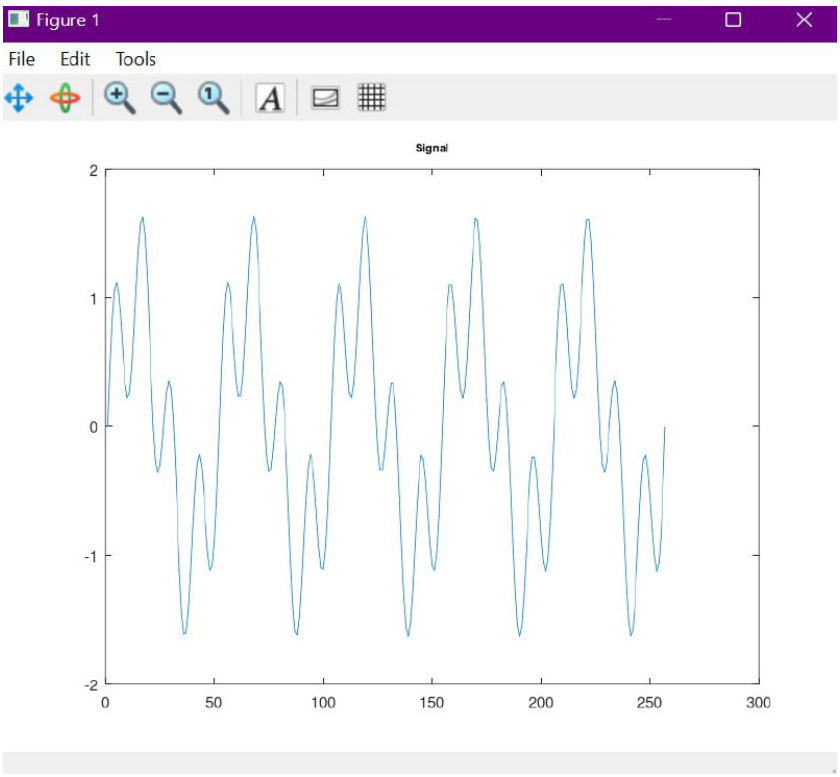


Рис. 2.16: Рисунок spectre_sum.m

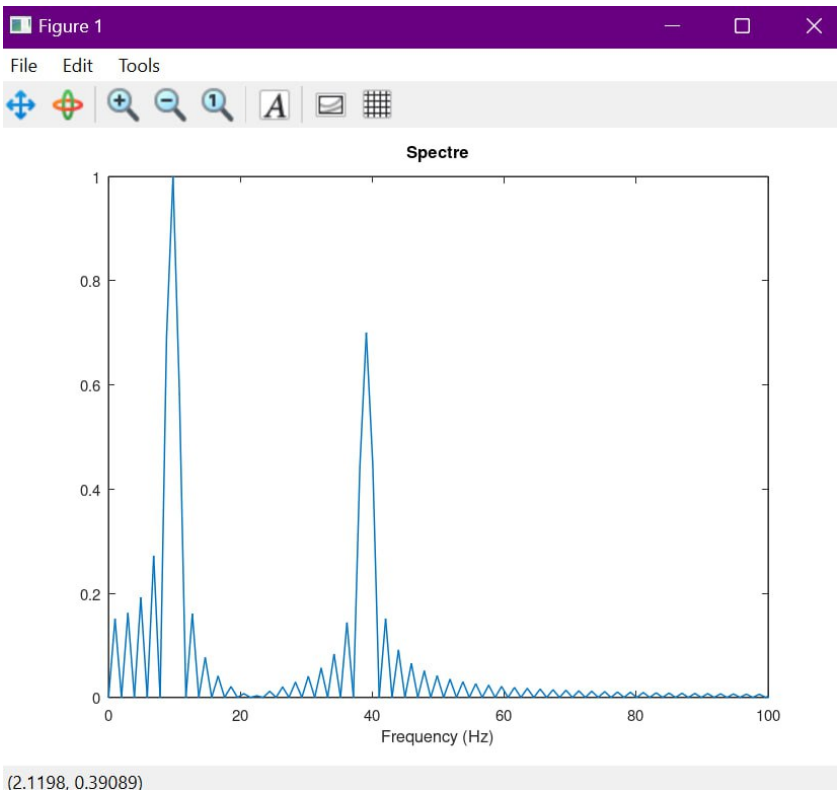


Рис. 2.17: Рисунок2 spectre_sum.m

9) Создал каталог modulation, а внутри создал файл am. Этот код нам показывает, что спектр произведения составляет свёртку спектров.

```
% 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]);
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]);
```

Рис. 2.18: Код am.m

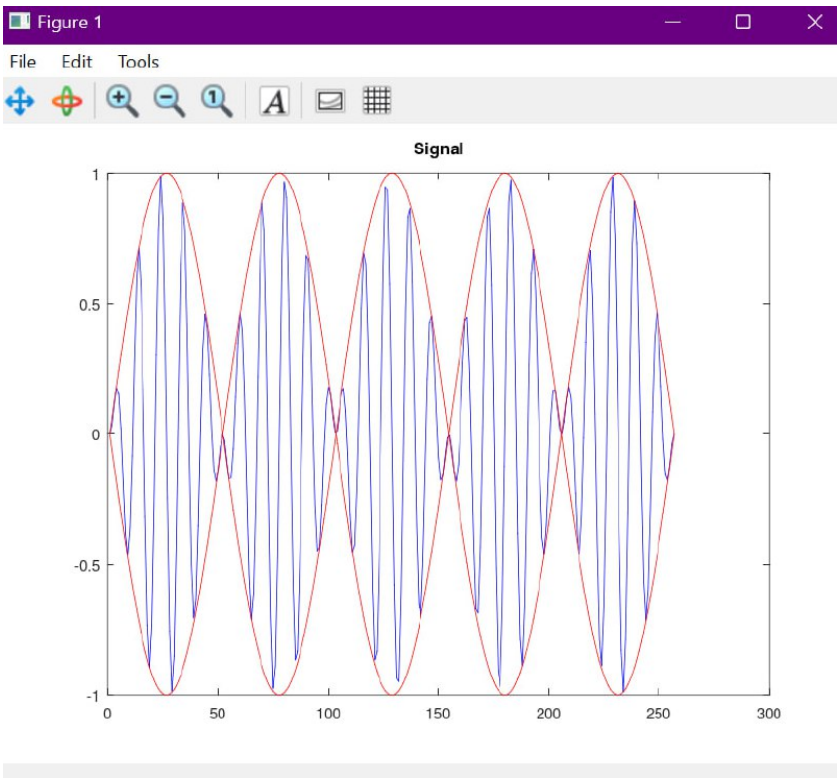


Рис. 2.19: Рисунок am.m

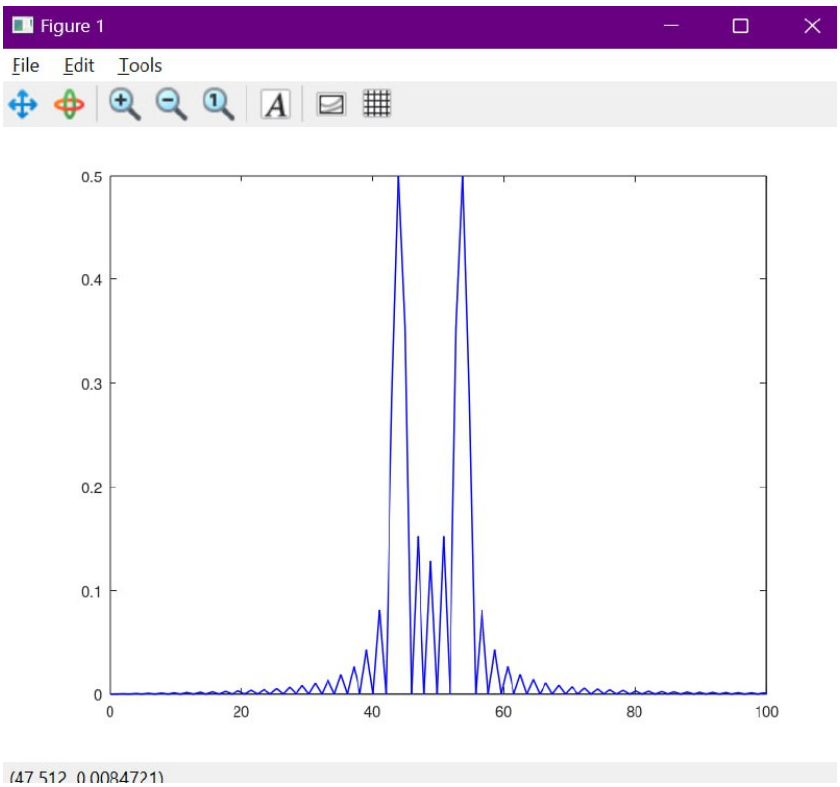


Рис. 2.20: Рисунок2 am.m

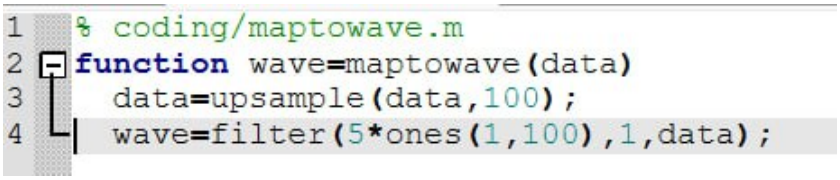
10) Я создал каталог coding. В нём создал файлы main.m, maptowave.m,unipolar.m,ami.m,bipola diffmanc.m, calcspectre.m. Проверил установлен ли пакет расширений.

Рис. 2.21: Проверка пакета расширений

11) Вписал код в каждый файл. Основной файл - main. С помощью него запускаются все остальные файлы.

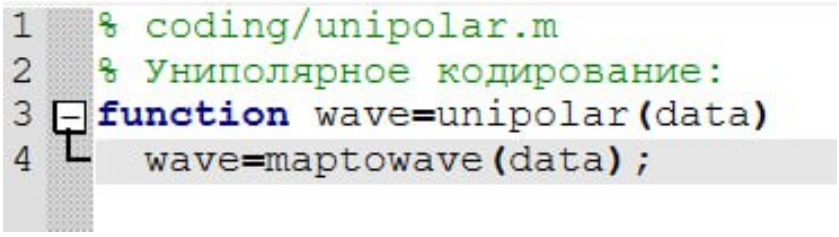
```
% coding/main.m
% Подключение пакета signal:
pkg load signal;
data=[0 1 0 0 1 1 0 0 0 1 1 0];
data_sync=[0 0 0 0 0 0 0 1 1 1 1 1 1 1];
data_spectre=[0 1 0 1 0 1 0 1 0 1 0 1 0 1];
mkdir 'signal';
mkdir 'sync';
mkdir 'spectre';
axis("auto");

% 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';
```

Рис. 2.22: Код main.m



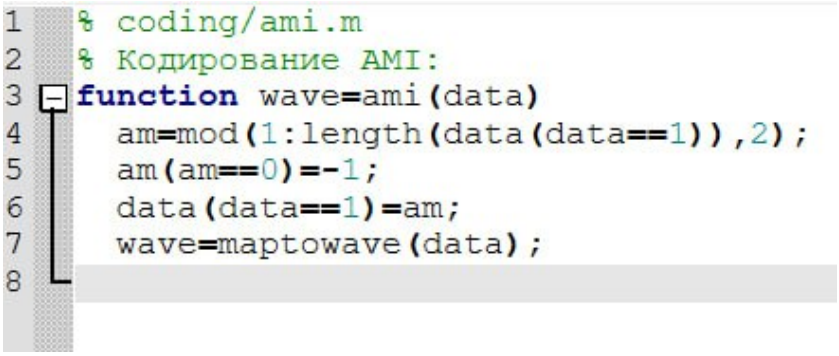
```
% coding/maptowave.m
function wave=maptowave(data)
  data=upsample(data,100);
  wave=filter(5*ones(1,100),1,data);
```

Рис. 2.23: Код maptowave.m



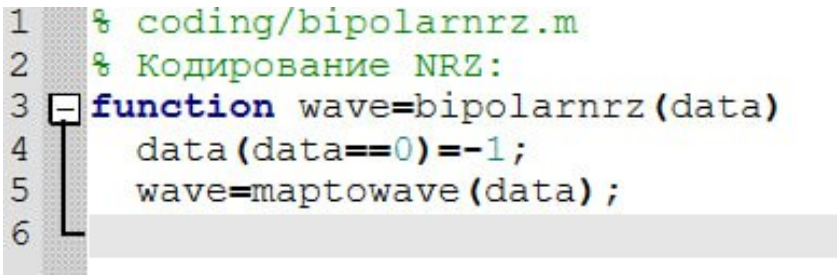
```
% coding/unipolar.m
% Униполярное кодирование:
function wave=unipolar(data)
  wave=maptowave(data);
```

Рис. 2.24: Код unipolar.m



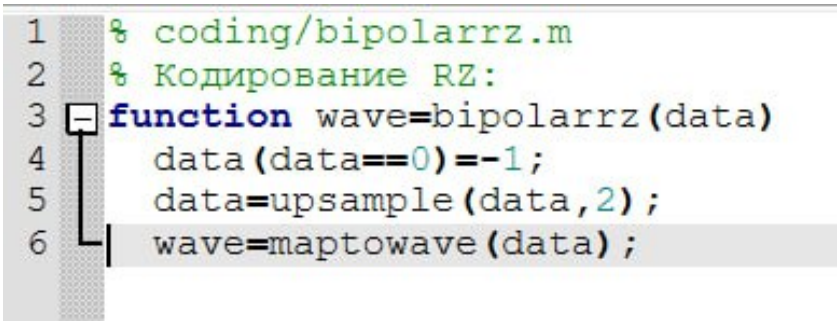
```
% coding/ami.m
% Кодирование AMI:
function wave=ami(data)
  am=mod(1:length(data(data==1)),2);
  am(am==0)=-1;
  data(data==1)=am;
  wave=maptowave(data);
```

Рис. 2.25: Код ami.m



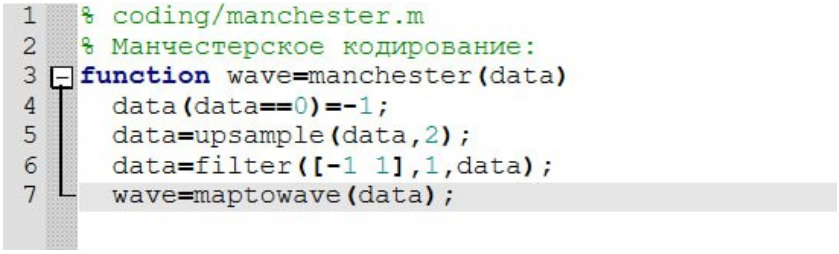
```
% coding/bipolarnrz.m
% Кодирование NRZ:
function wave=bipolarnrz(data)
  data(data==0)=-1;
  wave=maptowave(data);
```

Рис. 2.26: Код bipolarnrz.m



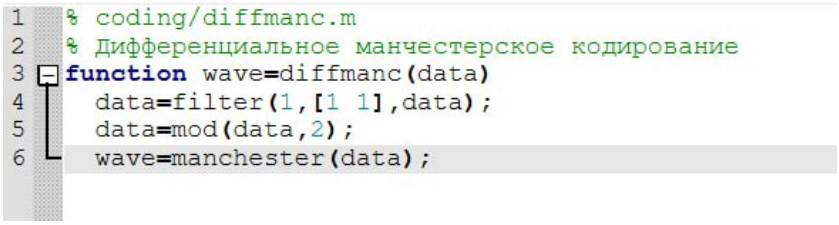
```
% coding/bipolarrz.m
% Кодирование RZ:
function wave=bipolarrz(data)
  data(data==0)=-1;
  data=upsample(data,2);
  wave=maptowave(data);
```

Рис. 2.27: Код bipolarrz.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);
```

Рис. 2.28: Код manchester.m



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

Рис. 2.29: Код diffmanc.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)');
```

Рис. 2.30: Код calcspectre.m

12) После запуска файла main у меня создались 3 каталога и создались несколько графиков, которые соответствуют каждому файлу.

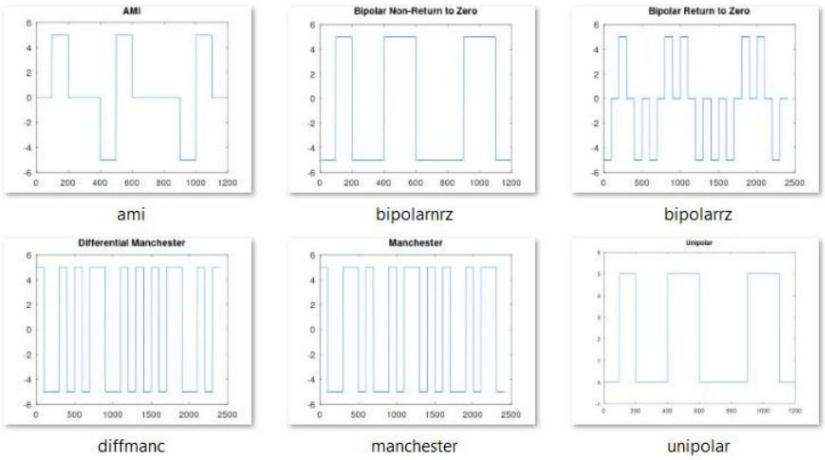


Рис. 2.31: Рисунки в каталоге signal

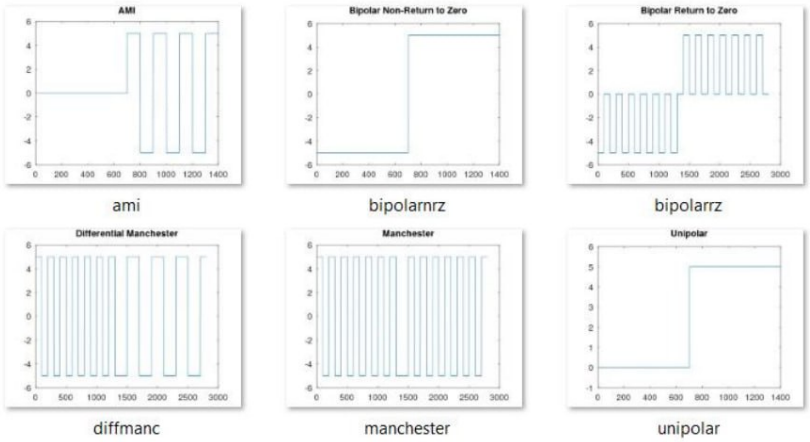


Рис. 2.32: Рисунки в каталоге sync

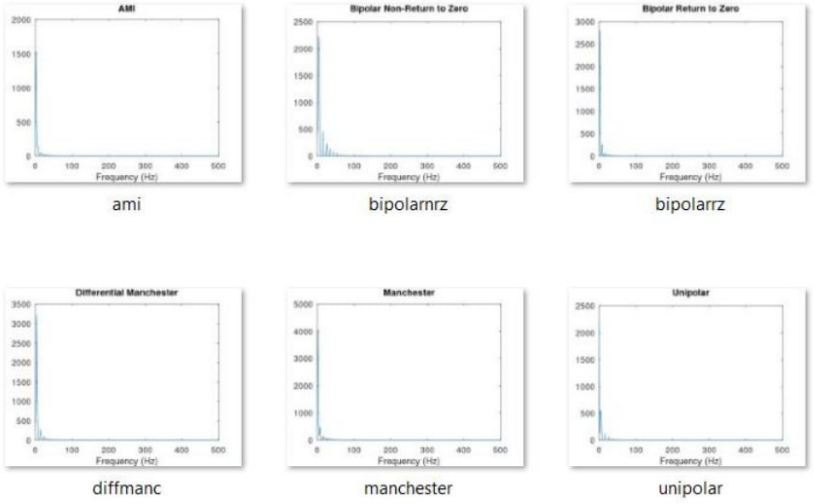


Рис. 2.33: Рисунки в каталоге spectre

3 Вывод

Я изучил методы кодирования и модуляции сигналов с помощью языка Octave. Определил спектры и параметры сигнала. Продемонстрировал принципы модуляции сигналов на примере аналоговой амплитудной модуляции. Исследовал свойства самосинхронизации сигнала.