Российский Университет Дружбы Народов

Факультет физико-математических и естественных наук.

Отчёт

о выполнении лабораторной работы № 1

по дисциплине: *сетевые технологии.*

Методы кодирования и модуляция сигналов

Студент: Танрибергенов Эльдар.

Группа: НПИбд-02-20

Студ. билет № 1032208074

Москва, 2022 г.

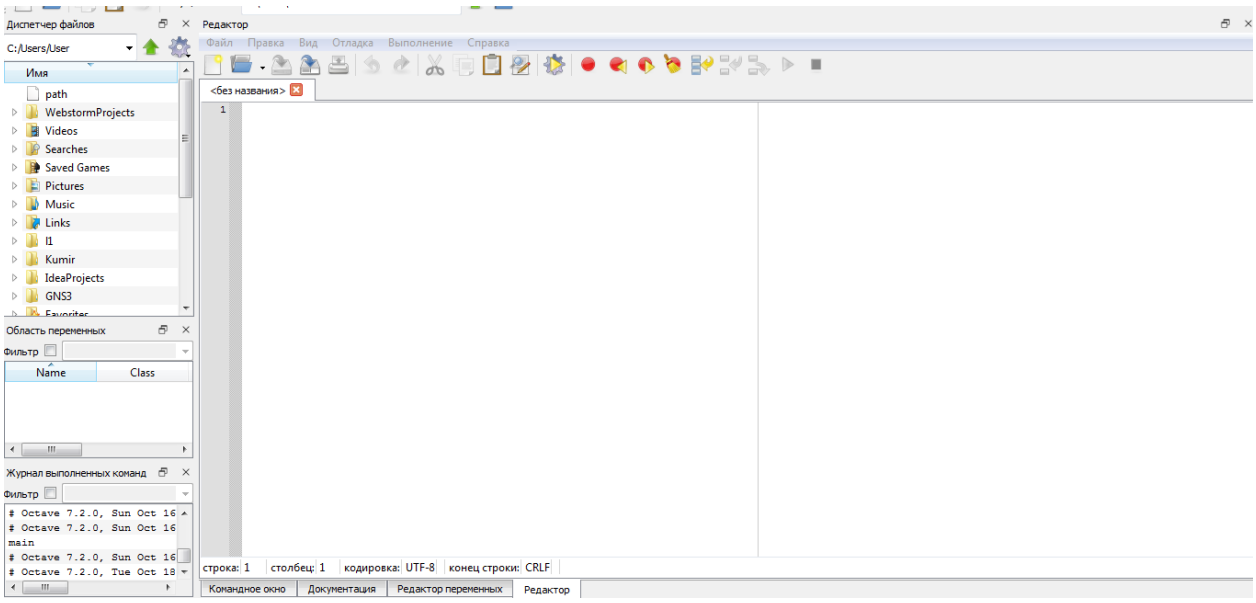
Цели работы:

- 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.

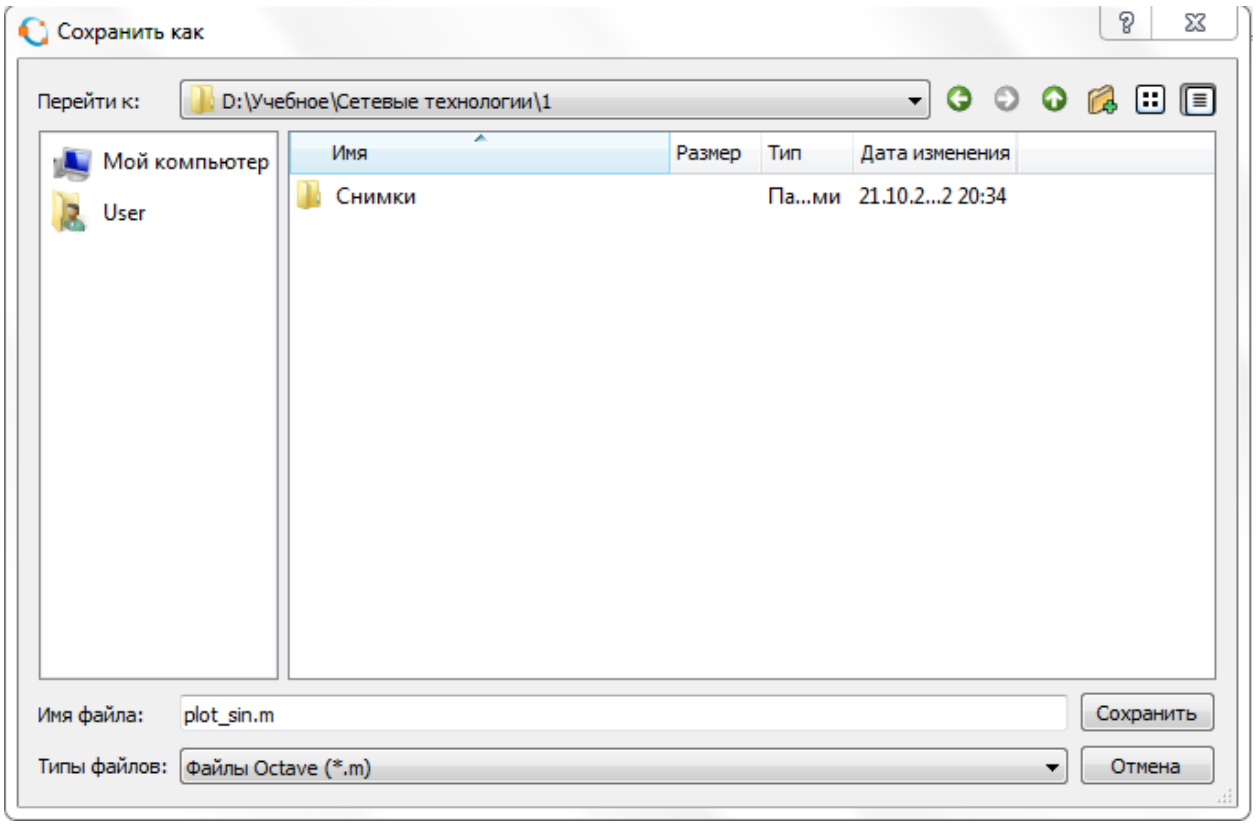
Ход работы:

1. Построение графиков в Octave.

Запустил Octave с оконным интерфейсом.

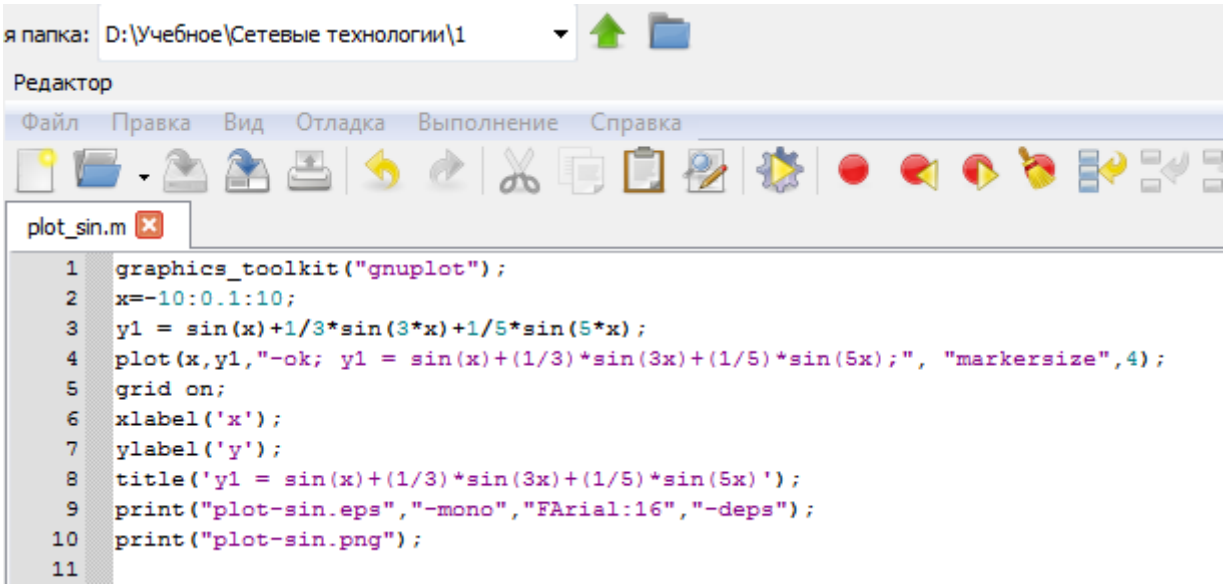


Перешёл в окно редактора. Воспользовавшись меню, создал новый сценарий. Сохранил его в рабочий каталог с именем plot_sin.m.

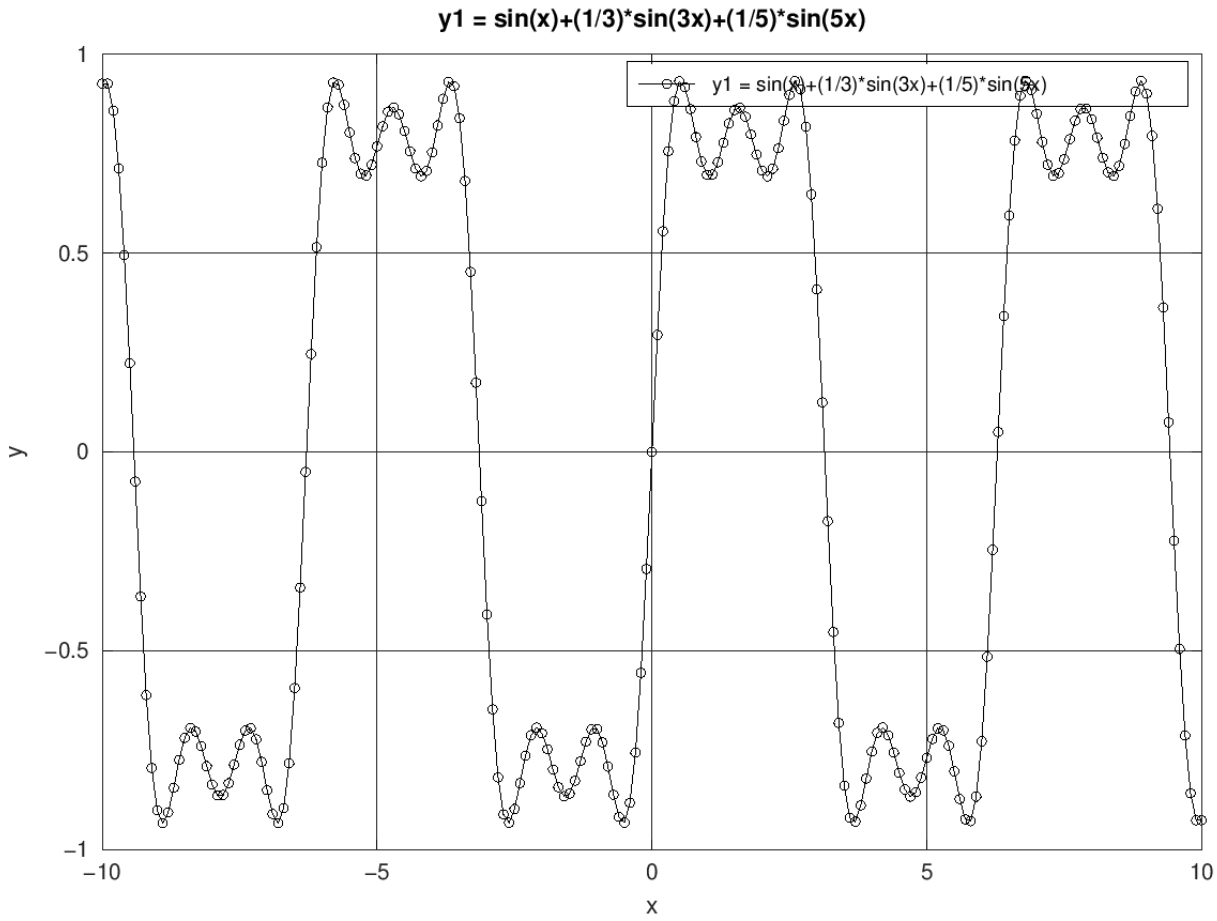


В окне редактора написал код для построения графика функции

$y = \sin(x) + \frac{1}{3}\sin(3x) + \frac{1}{5}\sin(5x)$ на интервале $[-10; 10]$:

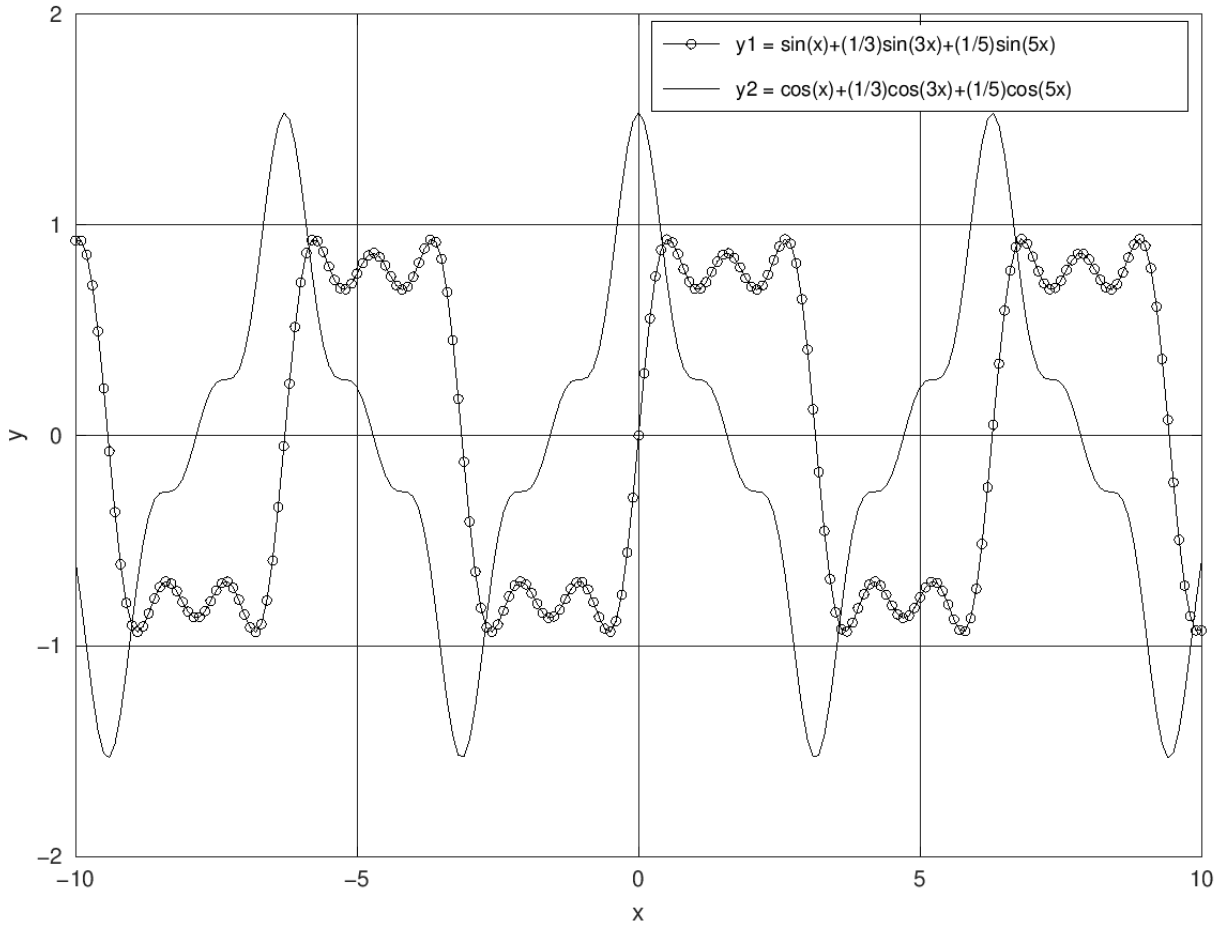


Запустил сценарий на выполнение. В качестве результата выполнения кода открылось окно с построенным графиком и в рабочем каталоге появились файлы с графиками в форматах .eps, .png.



Сохранил сценарий под другим названием и измените его так, чтобы на одном графике располагались отличающиеся по типу линий графики функций: предыдущий и

$$y = \cos(x) + \frac{1}{3}\cos(3x) + \frac{1}{5}\cos(5x)$$



2. Разложение импульсного сигнала в частичный ряд Фурье.

Разложение импульсного сигнала в форме меандра в частичный ряд Фурье можно задать формулой

$$s(t) = \frac{A}{2} + \frac{2A}{\pi}\left(\cos\left(\frac{2\pi}{T}t\right) - \frac{1}{3}\cos\left(3\frac{2\pi}{T}t\right) + \frac{1}{5}\cos\left(5\frac{2\pi}{T}t\right) - \cdots\right.$$

или формулой

$$s(t) = \frac{A}{2} + \frac{2A}{\pi}\left(\sin\left(\frac{2\pi}{T}t\right) + \frac{1}{3}\sin\left(3\frac{2\pi}{T}t\right) + \frac{1}{5}\sin\left(5\frac{2\pi}{T}t\right) + \cdots\right.$$

т.е. в спектре присутствуют только нечётные гармоники. Гармоники, образующие меандр, имеют амплитуду, обратно пропорциональную номеру соответствующей гармоники в спектре.

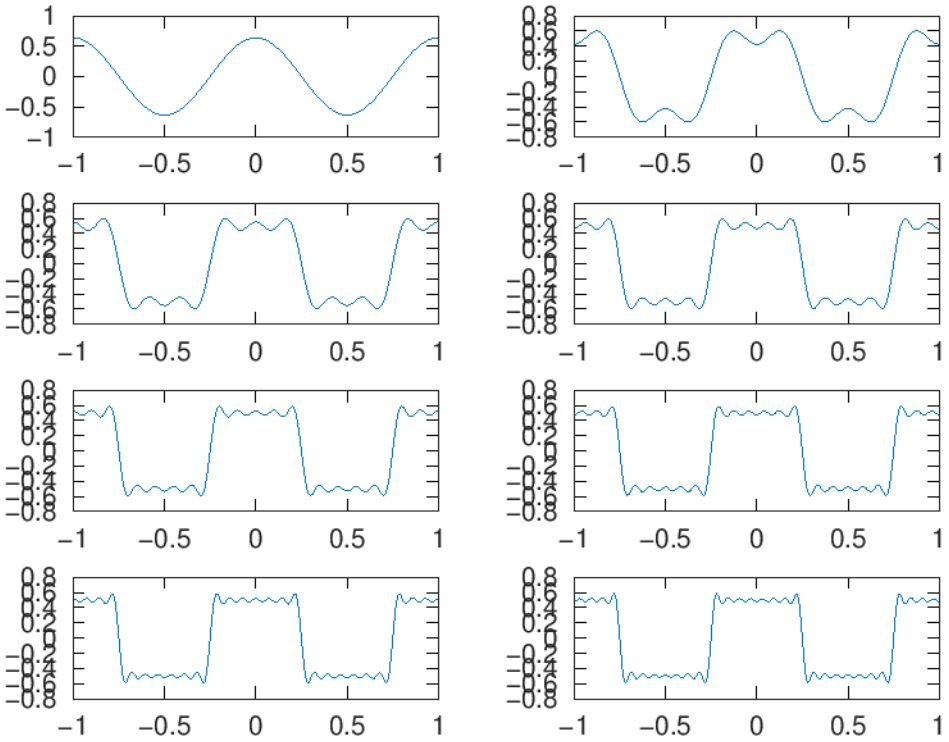
Создал новый сценарий и сохранил его в рабочий каталог с именем meandr.m следующего содержания:

meandr.m

```
graphics_toolkit("gnuplot");
N=8;
t=-1:0.01:1;
A=1;
T=1;
nh=(1:N)*2-1;
Am=2/pi ./ nh;
Am(2:2:end) = -Am(2:2:end);
harmonics=cos(2 * pi * nh' * t/T);
s1=harmonics.*repmat(Am',1,length(t));
s2=cumsum(s1);
for k=1:N
  subplot(4,2,k);
  plot(t, s2(k,:));
end

 print("plot-meandr.png");
```

Для построения в одном окне отдельных графиков меандра с различным количеством гармоник реализовал суммирование ряда с накоплением и воспользовался функциями subplot и plot для построения графиков. График экспортировал в файл png.

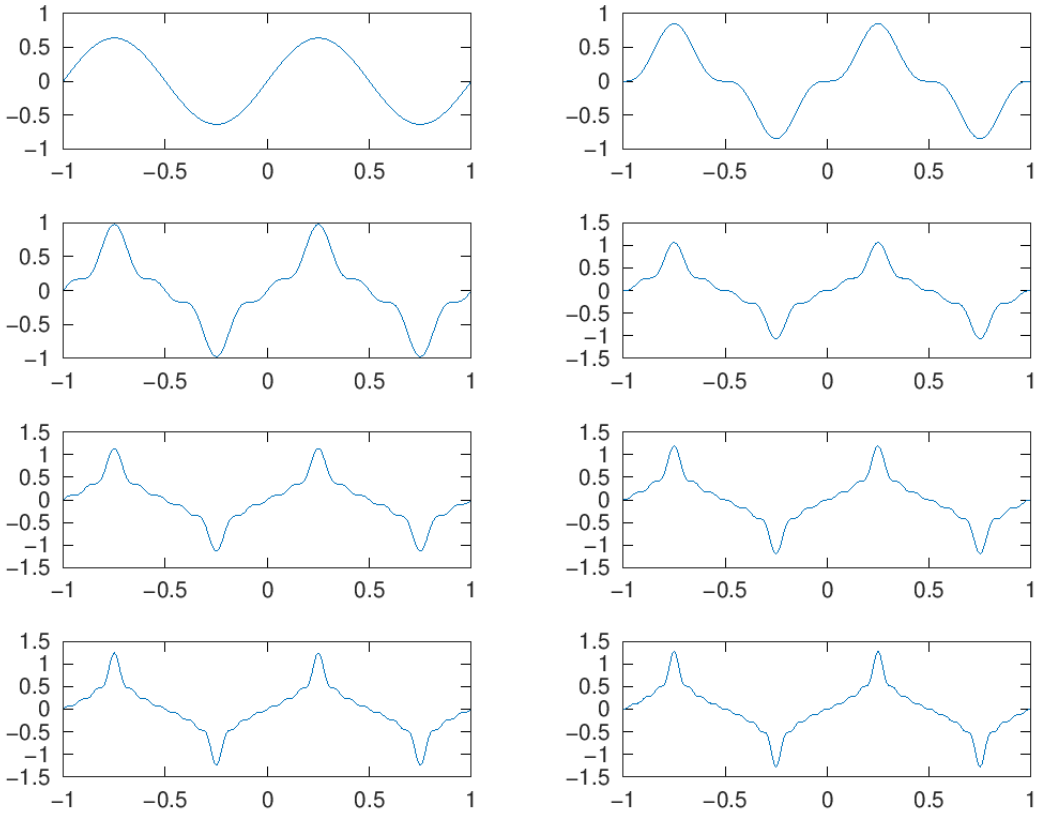
Результат:



Скорректировал для синусов:

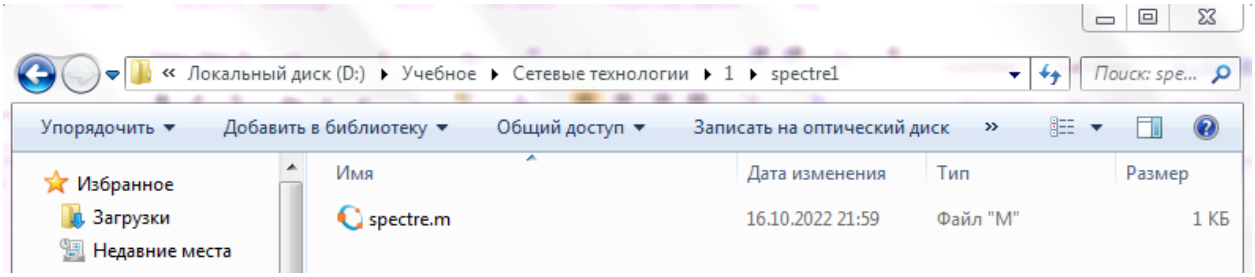
```
graphics_toolkit("gnuplot");
N=8;
t=-1:0.01:1;
A=1;
T=1;
nh=(1:N)*2-1;
Am=2/pi ./ nh;
Am(2:2:end) = -Am(2:2:end);
harmonics=sin(2 * pi * nh' * t/T);
s1=harmonics.*repmat(Am',1,length(t));
s2=cumsum(s1);
for k=1:N
  subplot(4,2,k);
  plot(t, s2(k,:));
end

 print("plot-meandr2.png");
```

Результат:



3. Определение спектра и параметров сигнала.

В рабочем каталоге создал каталог spectre1 и в нём новый сценарий с именем spectre.m.

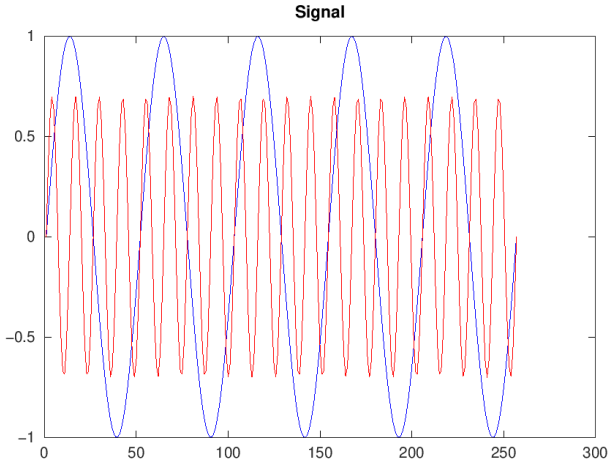


Код:

spectre.m

```
1  graphics_toolkit("gnuplot");
2  mkdir 'signal';
3  mkdir 'spectre';
4  tmax = 0.5;
5  fd = 512;
6  f1 = 10;
7  f2 = 40;
8  a1 = 1;
9  a2 = 0.7;
10 t = 0:1./fd:tmax;
11 fd2 = fd/2;
12 signal1 = a1*sin(2*pi*t*f1);
13 signal2 = a2*sin(2*pi*t*f2);
14 plot(signal1,'b');
15 hold on
16 plot(signal2,'r');
17 hold off
18 title('Signal');
19 print 'signal/spectre.png';
```

Результат:

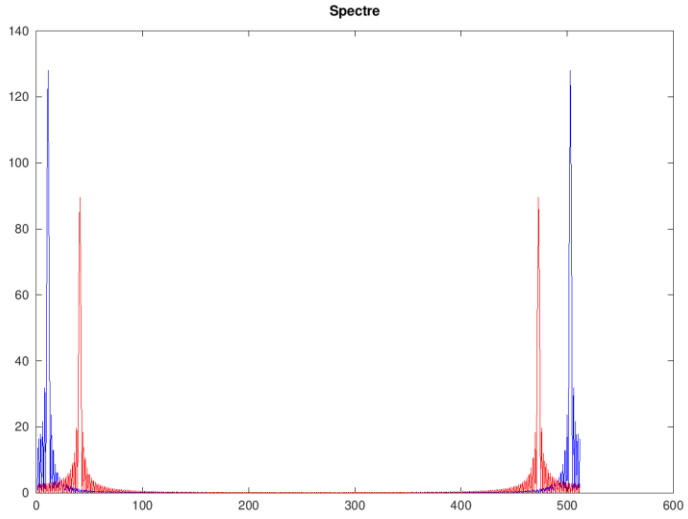


С помощью быстрого преобразования Фурье нашёл спектры сигналов, добавив новый код.

Код:

```
spectre1 = abs(fft(signal1,fd));
spectre2 = abs(fft(signal2,fd));
plot(spectre1,'b');
hold on
plot(spectre2,'r');
hold off
title('Spectre');
print 'spectre/spectre.png';
```

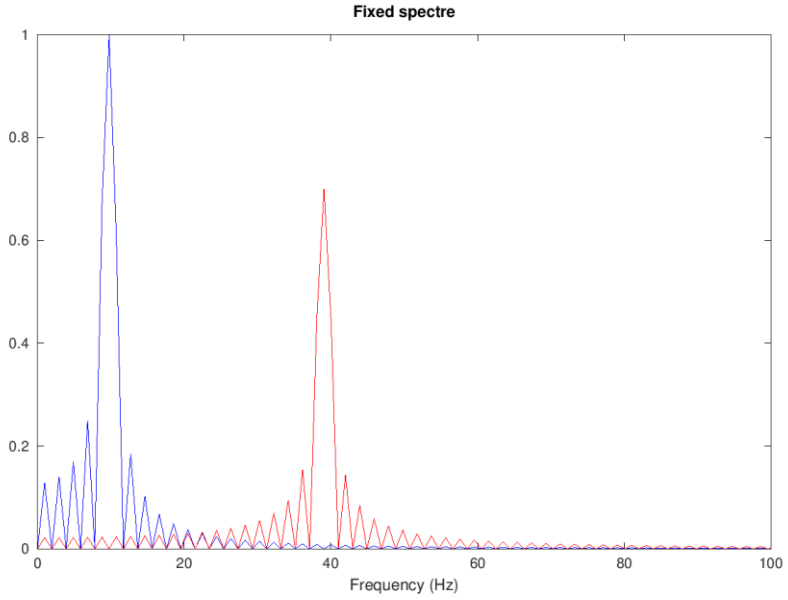
Результат:



Учитывая реализацию преобразования Фурье, скорректировал график спектра. Отбросил дублирующие отрицательные частоты, а также принял в расчёт то, что на каждом шаге вычисления быстрого преобразования Фурье происходит суммирование амплитуд сигналов. Для этого добавил в файл spectre.m следующий код:

```
f = 1000*(0:fd2)./(2*fd);
spectre1 = 2*spectre1/fd2;
spectre2 = 2*spectre2/fd2;
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';
```

Результат:

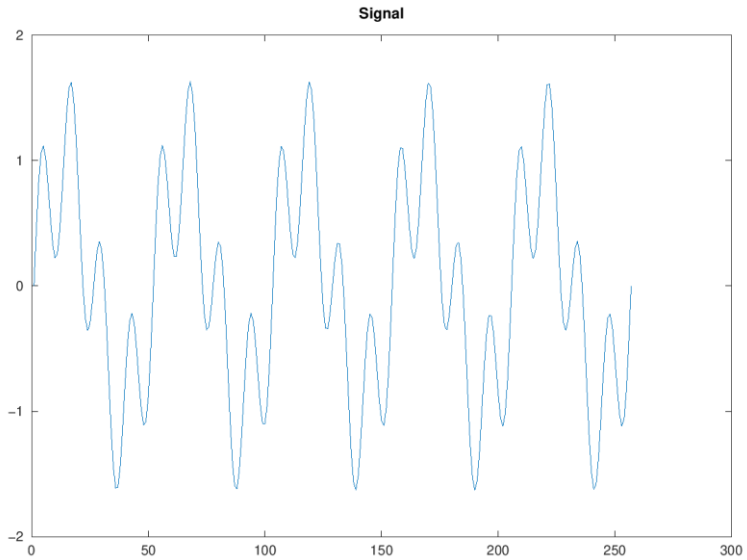


Нашёл спектр суммы рассмотренных сигналов, создав каталог spectr_sum и файл в нём spectre_sum.m со следующим кодом:

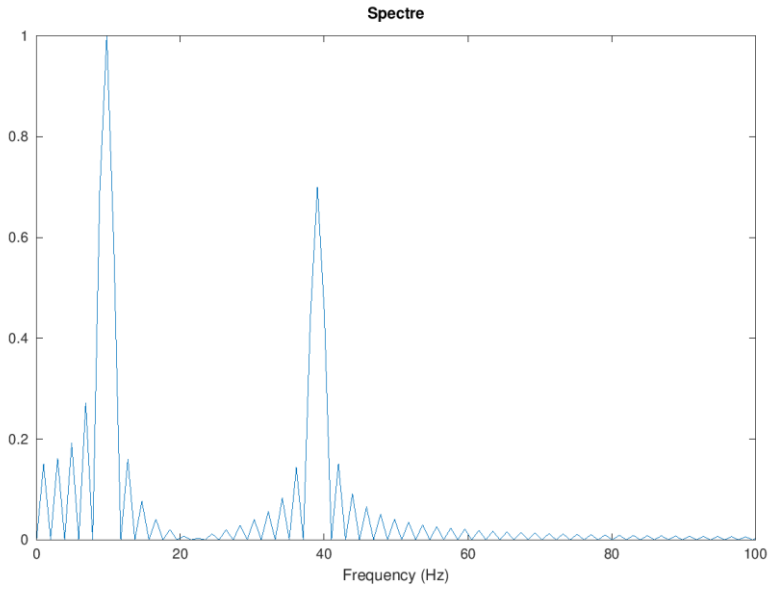
spectre_sum.m

```
graphics_toolkit("gnuplot");
mkdir 'signal';
mkdir 'spectre';
tmax = 0.5;
fd = 512;
f1 = 10;
f2 = 40;
a1 = 1;
a2 = 0.7;
fd2 = fd/2;
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';
spectre = fft(signal,fd);
f = 1000*(0:fd2)./(2*fd);
spectre = 2*sqrt(spectre.*conj(spectre))./fd2;
plot(f,spectre(1:fd2+1))
xlim([0 100]);
title('Spectre');
xlabel('Frequency (Hz)');
print 'spectre/spectre_sum.png';
```

Результат:



Суммарный сигнал



Спектр суммарного сигнала.

4. Амплитудная модуляция.

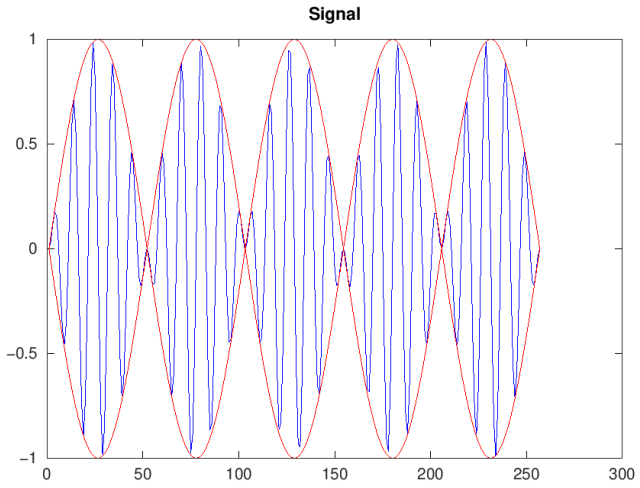
В рабочем каталоге создал каталог modulation и в нём новый сценарий с именем am.m.

Код:

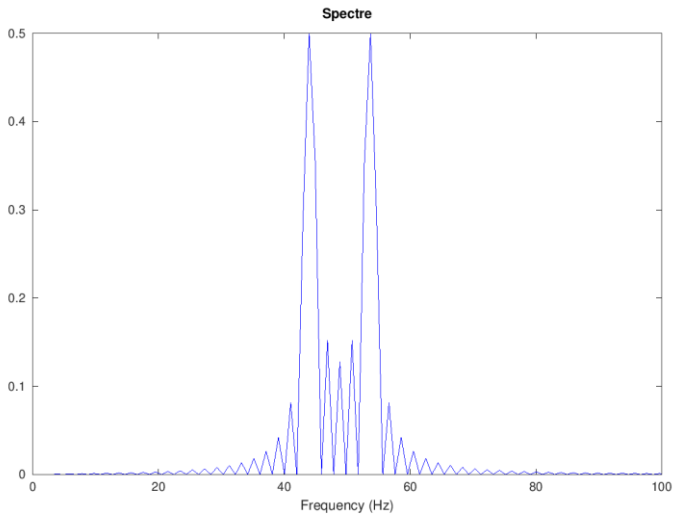
am.m

```
graphics_toolkit("gnuplot");
mkdir 'signal';
mkdir 'spectre';
tmax=0.5;
fd=512;
f1=5;
f2=50;
fd2 = fd/2;
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;
plot(signal1,'r');
plot(-signal1,'r');
hold off;
title('Signal');
print 'signal/am.png';
spectre = fft(signal,fd);
f = 1000*(0:fd2)./(2*fd);
spectre = 2*sqrt(spectre.*conj(spectre))./fd2;
plot(f,spectre(1:fd2+1), 'b');
xlim([0 100]);
title('Spectre');
xlabel('Frequency (Hz)');
print 'spectre/am.png';
```

В результате получаем, что спектр произведения представляет собой свёртку спектров



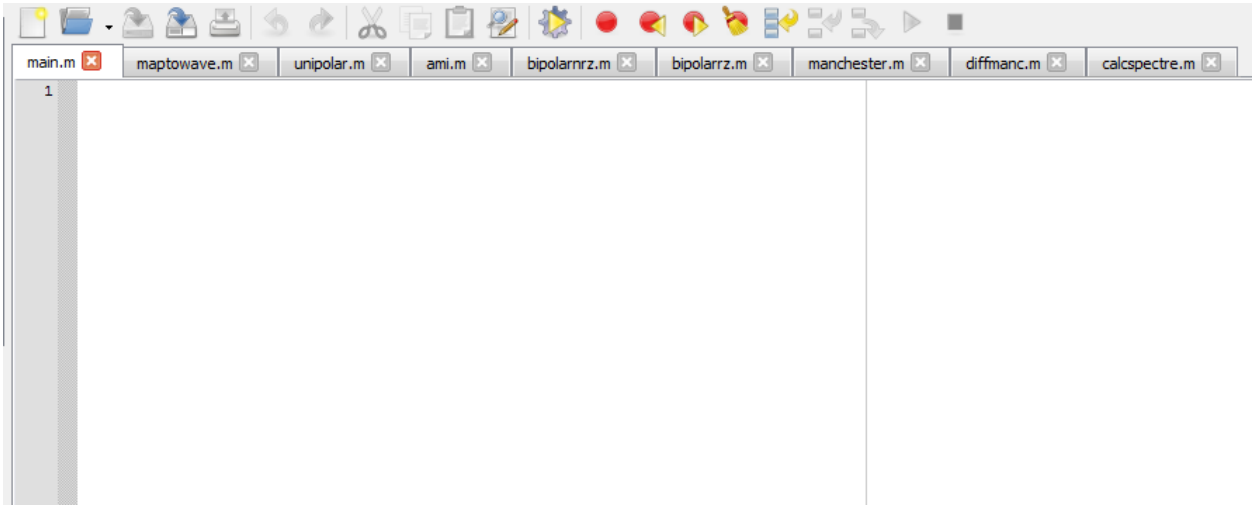
Сигнал и огибающая при амплитудной модуляции



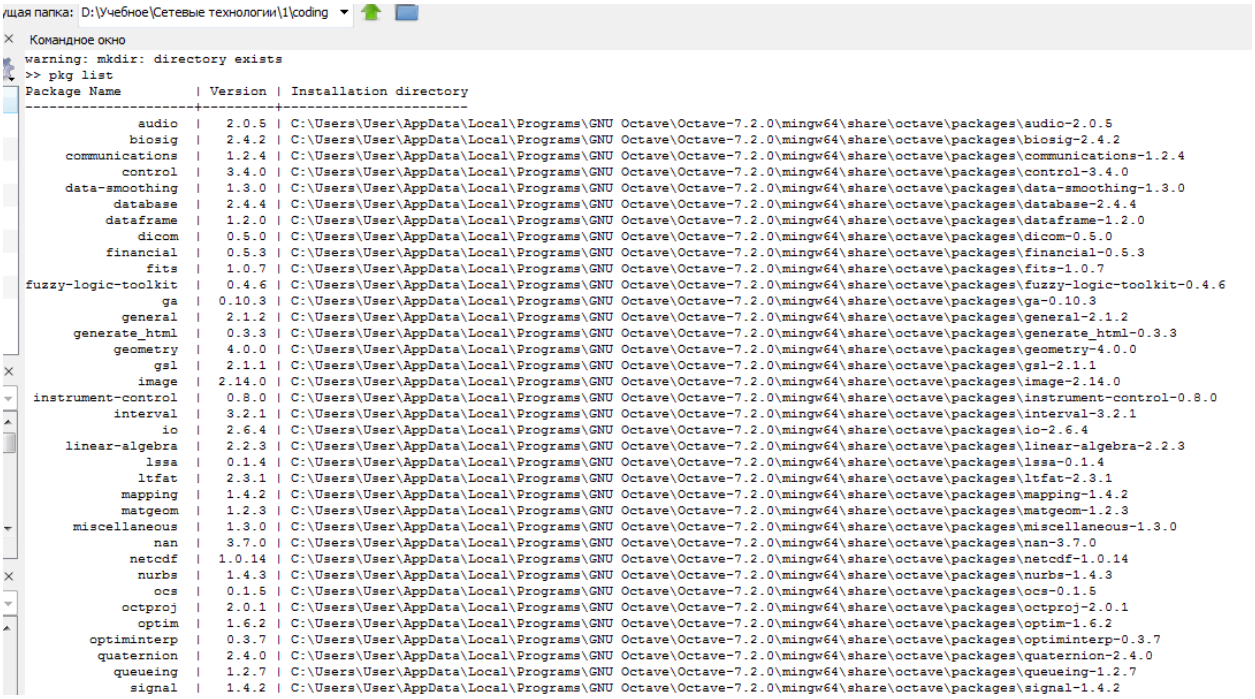
Спектр сигнала при амплитудной модуляции

5. Кодирование сигнала. Исследование самосинхронизации сигнала.

В рабочем каталоге создал каталог coding и в нём файлы main.m, maptowave.m, unipolar.m, ami.m,bipolarnrz.m,bipolarrz.m,manchester.m, diffmanc.m, calcspectre.m.



В окне интерпретатора команд проверил, установлен ли пакет расширений signal – установлен.



В файле main.m подключил пакет signal и задал входные кодовые последовательности

main.m

```
graphics_toolkit("gnuplot");
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");
```

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

```
wave=unipolar(data);
plot(wave);
ylim([-1 6]);
title('Unipolar');
print 'signal/unipolar.png';

wave=ami(data);
plot(wave)
title('AMI');
print 'signal/ami.png';

wave=bipolarnrz(data);
plot(wave);
title('Bipolar Non-Return to Zero');
print 'signal/bipolarnrz.png';

wave=bipolarrz(data);
plot(wave)
title('Bipolar Return to Zero');
print 'signal/bipolarrz.png';

wave=manchester(data);
plot(wave)
title('Manchester');
print 'signal/manchester.png';

wave=diffmanc(data);
plot(wave)
title('Differential Manchester');
print 'signal/diffmanc.png';
```

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

```
wave=unipolar(data_sync);
plot(wave);
ylim([-1 6]);
title('Unipolar');
print 'sync/unipolar.png';

wave=ami(data_sync);
plot(wave)
title('AMI');
print 'sync/ami.png';

wave=bipolarnrz(data_sync);
plot(wave);
title('Bipolar Non-Return to Zero');
print 'sync/bipolarnrz.png';

wave=bipolarrz(data_sync);
plot(wave)
title('Bipolar Return to Zero');
print 'sync/bipolarrz.png';

wave=manchester(data_sync);
plot(wave)
title('Manchester');
print 'sync/manchester.png';

wave=diffmanc(data_sync);
plot(wave)
title('Differential Manchester');
print 'sync/diffmanc.png';
```

Далее в этом же файле прописал вызовы функций для построения графиков спектров:

```
wave=unipolar(data_spectre);
spectre=calcspectre(wave);
title('Unipolar');
print 'spectre/unipolar.png';

wave=ami(data_spectre);
spectre=calcspectre(wave);
title('AMI');
print 'spectre/ami.png';

wave=bipolarnrz(data_spectre);
spectre=calcspectre(wave);
title('Bipolar Non-Return to Zero');
print 'spectre/bipolarnrz.png';

wave=bipolarrz(data_spectre);
spectre=calcspectre(wave);
title('Bipolar Return to Zero');
print 'spectre/bipolarrz.png';

wave=manchester(data_spectre);
spectre=calcspectre(wave);
title('Manchester');
print 'spectre/manchester.png';

wave=diffmanc(data_spectre);
spectre=calcspectre(wave);
title('Differential Manchester');
print 'spectre/diffmanc.png';
```

В файле maptowave.m прописал функцию, которая по входному битовому потоку строит график сигнала

```
function wave=maptowave(data)
    data=upsample(data,100);
    wave=filter(5*ones(1,100),1,data);
```

В файлах unipolar.m, ami.m, bipolarnrz.m, bipolarrz.m, manchester.m, diffmanc.m прописал соответствующие функции преобразования кодовой последовательности data с вызовом функции maptowave для построения соответствующего графика.

Униполярное кодирование:

```
function wave=unipolar(data)
    wave=maptowave(data);

```

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);
```

Кодирование NRZ:

```
function wave=bipolarnrz(data)
  data(data==0)=-1;
  wave=maptowave(data);
```

Кодирование RZ:

```
function wave=bipolarrz(data)
  data(data==0)=-1;
  data=upsample(data,2);
  wave=maptowave(data);
```

Манчестерское кодирование:

```
function wave=manchester(data)
  data(data==0)=-1;
  data=upsample(data,2);
  data=filter([-1 1],1,data);
  wave=maptowave(data);
```

Дифференциальное манчестерское кодирование:

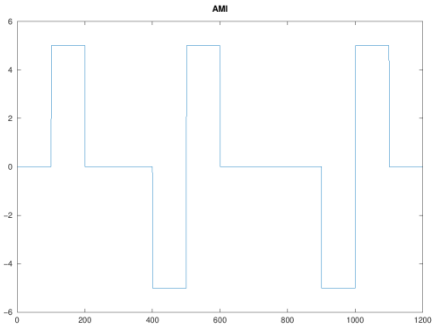
```
function wave=diffmanc(data)
  data=filter(1,[1 1],data);
  data=mod(data,2);
  wave=manchester(data);
```

В файле calcspectre.m прописал функцию построения спектра сигнала

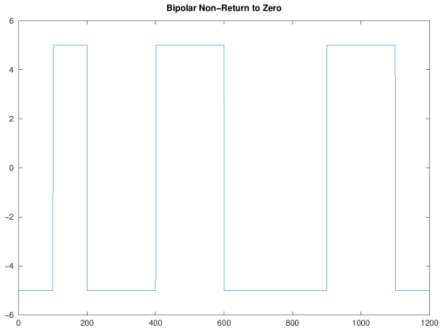
```
function spectre = calcspectre(wave)
  Fd = 512;
  Fd2 = Fd/2;
  Fd3 = Fd/2 + 1;

  X = fft(wave,Fd);
  spectre = X.*conj(X)/Fd;
  f = 1000*(0:Fd2)/Fd;
  plot(f,spectre(1:Fd3));
  xlabel('Frequency (Hz)');
```

Запустил 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.

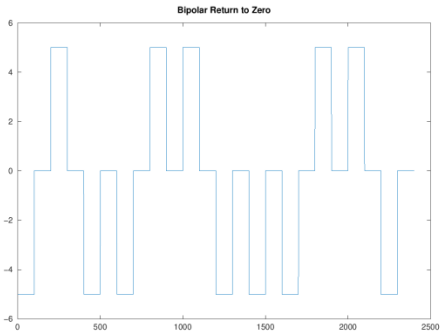
Графики кодированного сигнала:



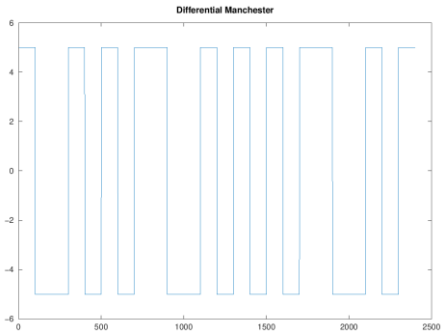
AMI



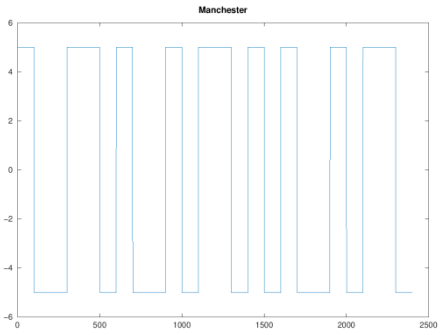
NRZ



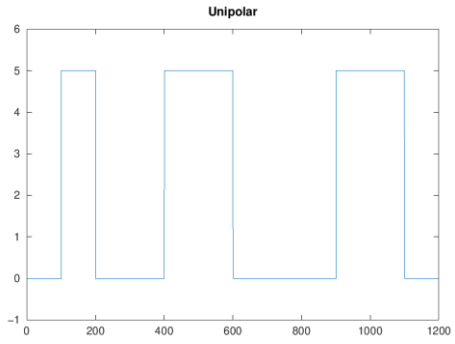
RZ



Дифференциальное манчестерское

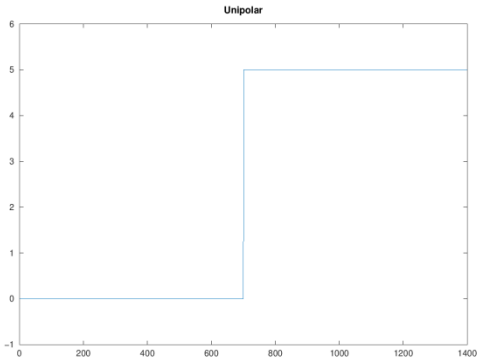


Манчестерское

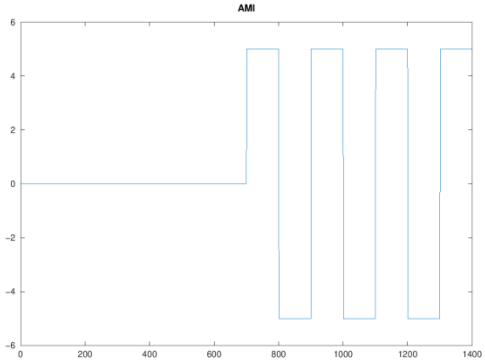


Униполярное

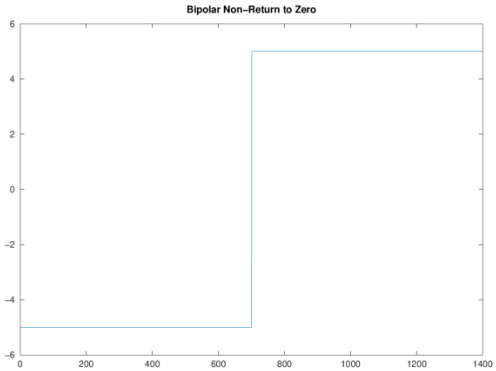
Графики свойств самосинхронизации:



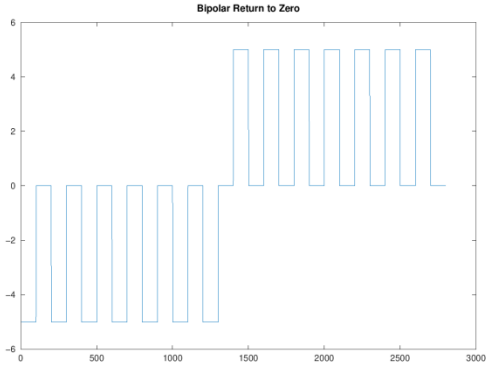
Униполярное кодирование:
нет самосинхронизации



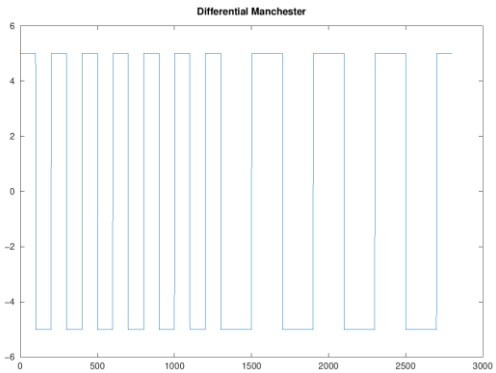
Кодирование AMI:
самосинхронизация при наличии сигнала



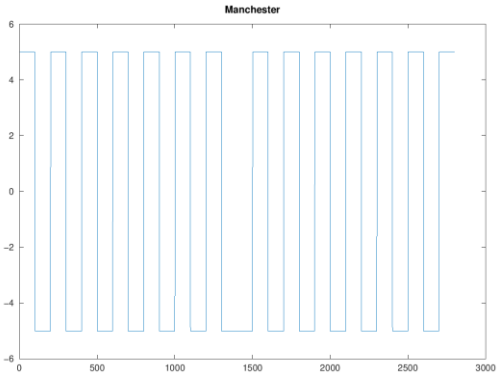
Кодирование NRZ:
нет самосинхронизации



Кодирование RZ:
есть самосинхронизация

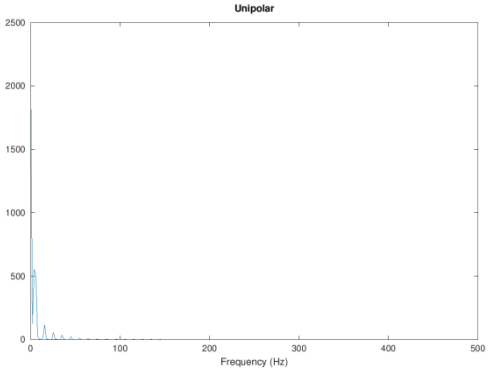


Дифференциальное манчестерское
кодирование: есть самосинхронизация

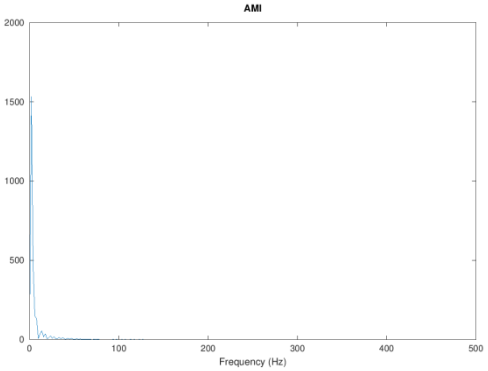


Манчестерское кодирование:
есть самосинхронизация

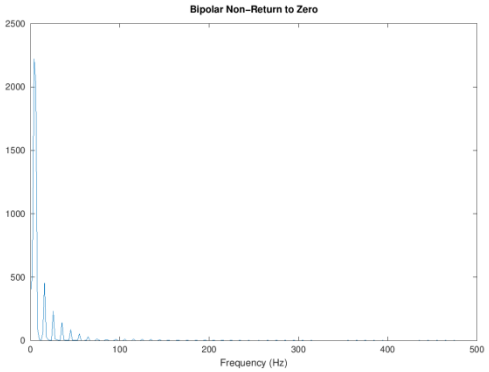
Графики спектров сигнала:



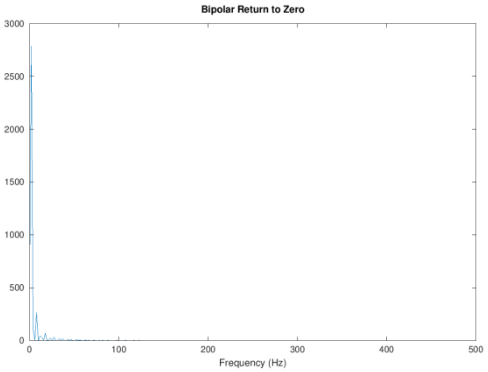
Униполярное кодирование



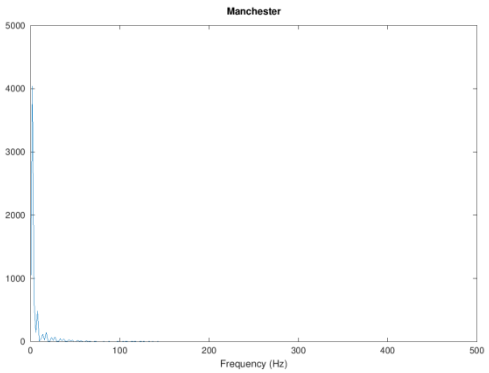
Кодирование AMI



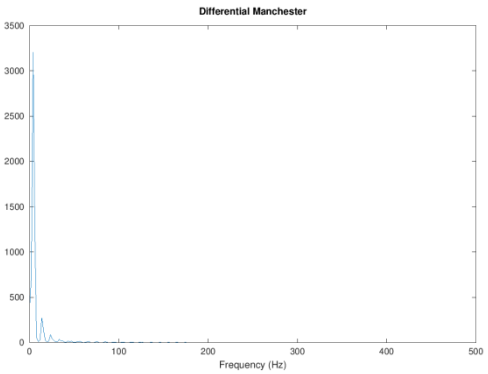
Кодирование NRZ



Кодирование RZ



Манчестерское кодирование



Дифференциальное манчестерское

Вывод:

В результате лабораторной работы я изучил методы кодирования и модуляции сигналов с помощью высокоуровнего языка программирования Octave, а также освоил определения спектра и параметров сигнала, принципы модуляции сигнала на примере аналоговой амплитудной модуляции. Исследовал свойства самосинхронизации сигнала.