

Российский Университет Дружбы Народов

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

Презентация

выполненной 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 высокоуровневого 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.

1.1. Запуск Octave. Создание нового сценария в окне редактора. Сохранение с именем plot_sin.m (рис. 1.1.1 – 1.1.2).

Использовано:

- Программа Octave;
- Графический пользовательский интерфейс.

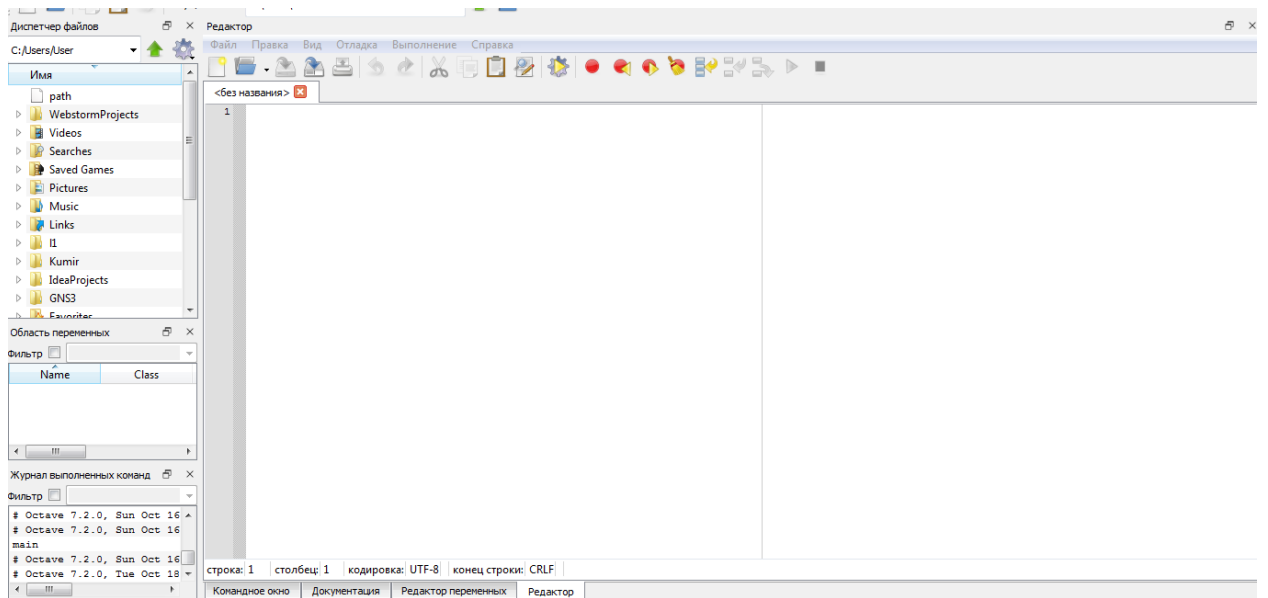


Рис. 1.1.1. Запуск Octave.

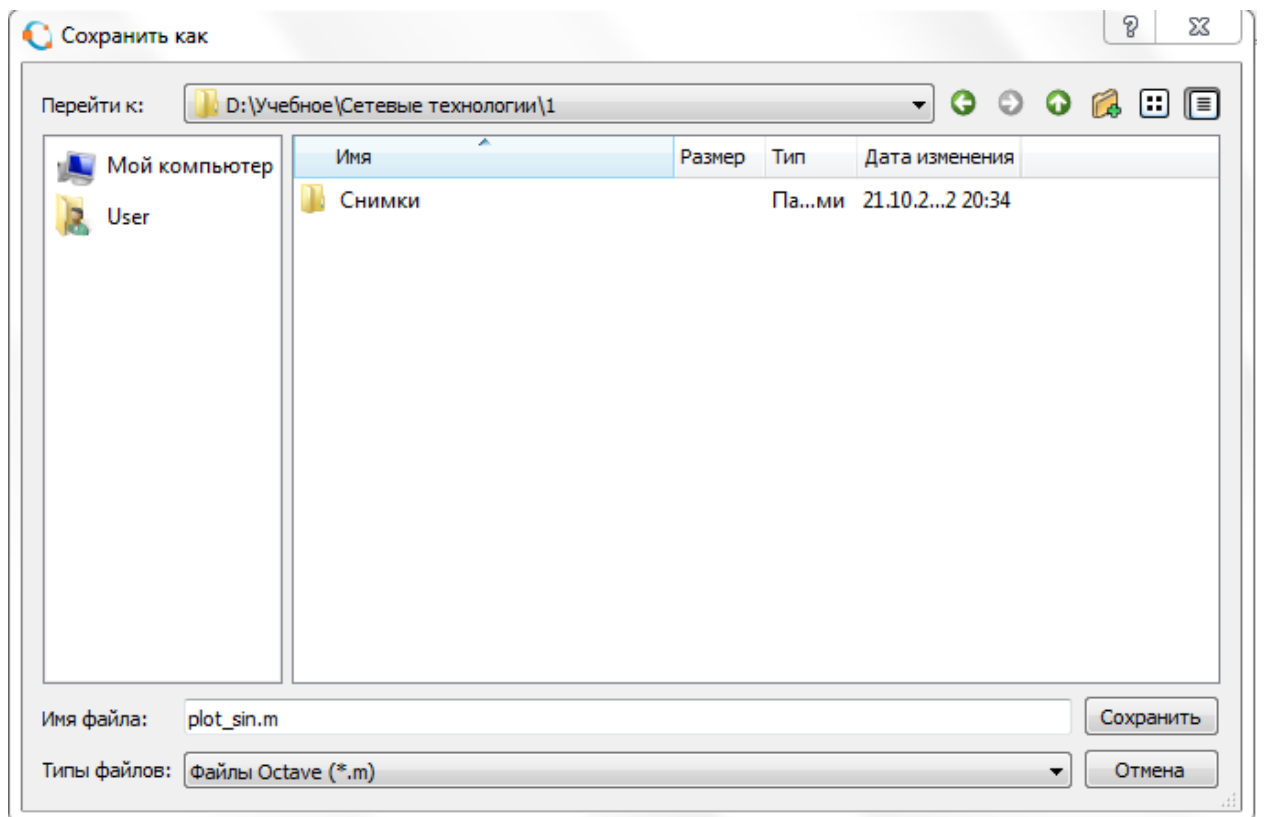


Рис. 1.1.2. Сохранение сценария.

1.2. Построение графика функции $y_1 = \sin(x) + \frac{1}{3}\sin(3x) + \frac{1}{5}\sin(5x)$ на интервале $[-10; 10]$ (рис. 1.2.1 – 1.2.2).

Использовано:

- Формула функции из задания;
- Функция plot.

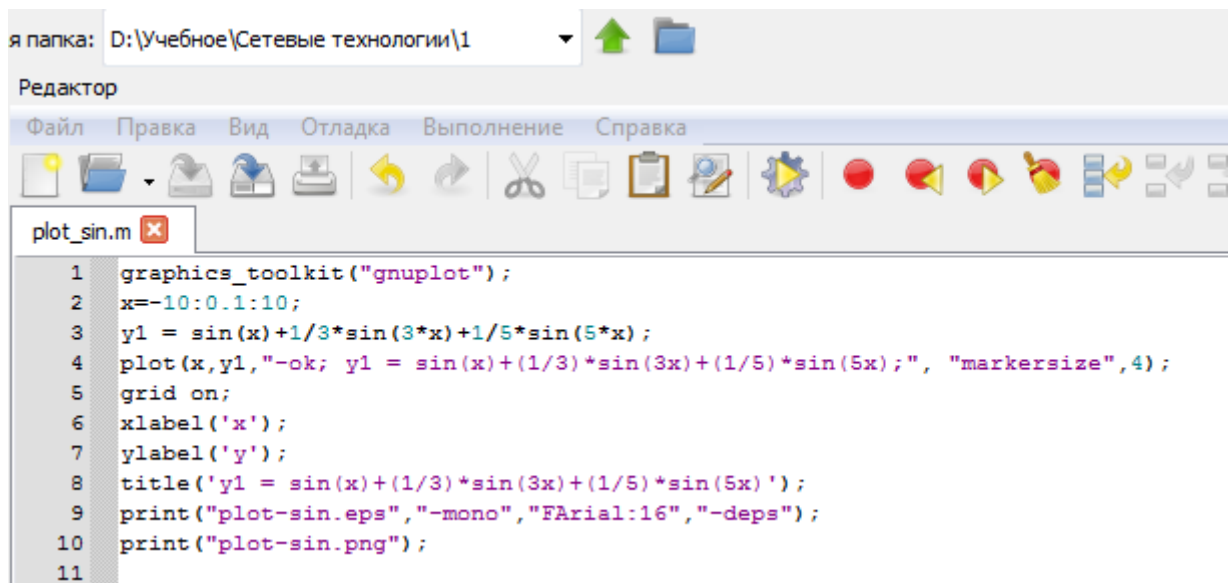


Рис. 1.2.1. Код для построения графика функции.

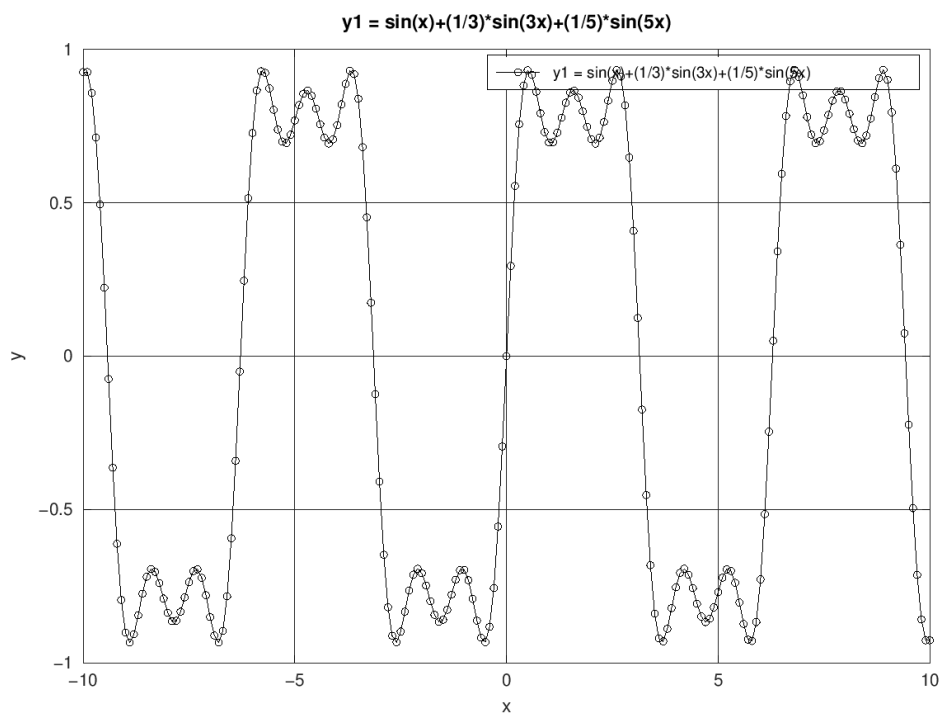


Рис. 1.2.2. График функции.

1.3. Добавление графика функции $y_2 = \cos(x) + \frac{1}{3}\cos(3x) + \frac{1}{5}\cos(5x)$
(рис. 1.3.1 – 1.3.2).

Использовано:

- Формулы функций из задания;
- Функция plot.

```
1 graphics_toolkit("gnuplot");
2 x=-10:0.1:10;
3 y1 = sin(x)+1/3*sin(3*x)+1/5*sin(5*x);
4 y2 = cos(x)+1/3*cos(3*x)+1/5*cos(5*x);
5 plot(x,y1,"-o", y1 = sin(x)+(1/3)sin(3x)+(1/5)sin(5x);", "markersize",4,x,y2,"-k", y2 = cos(x)+(1/3)cos(3x)+(1/5)cos(5x);", "markersize",4);
6 grid on;
7 xlabel('x');
8 ylabel('y');
9 print("plot-sin_cos.eps","-mono","Arial:16","-deps");
10 print("plot-sin_cos.png");
11
```

Рис. 1.3.1. Код.

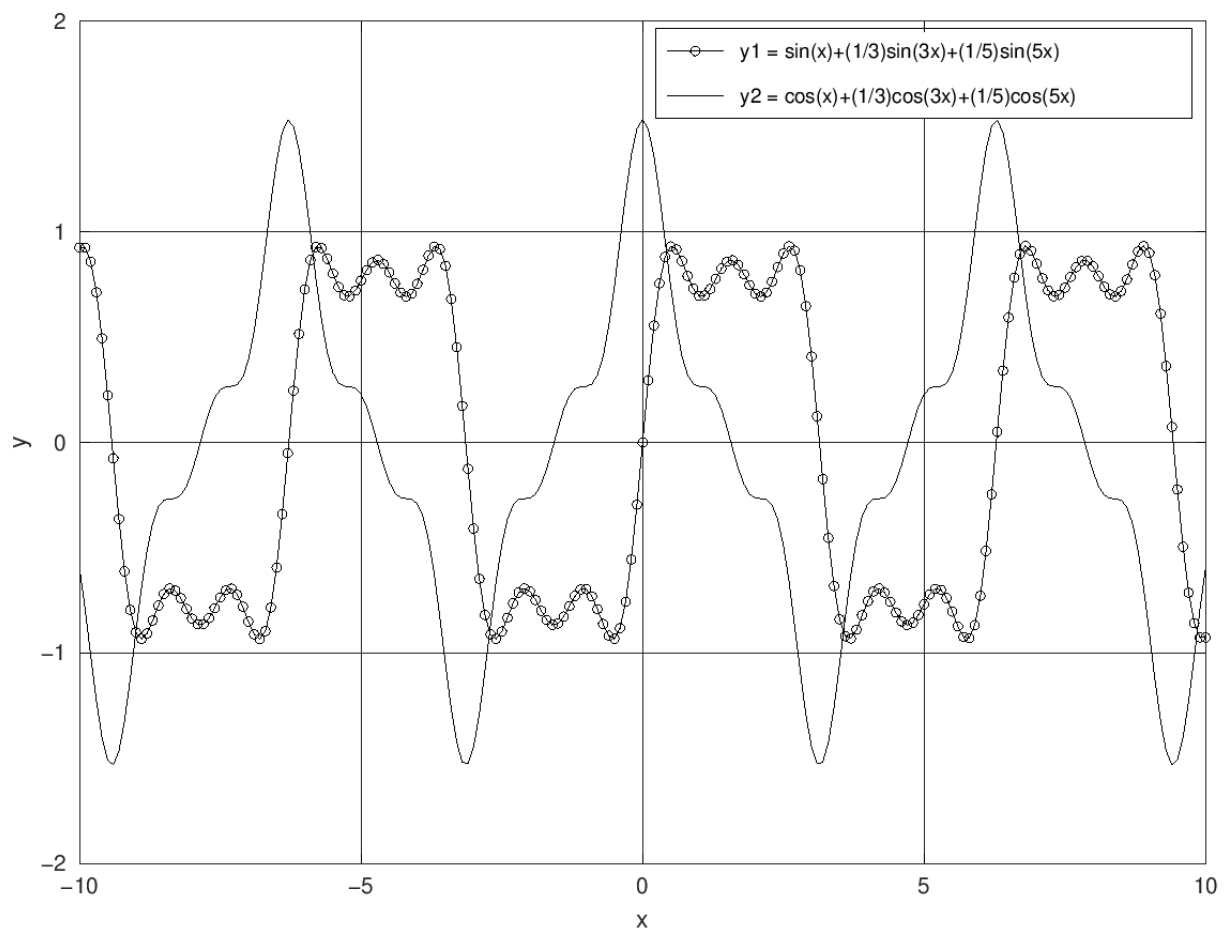


Рис. 1.3.2. График функций.

2. Разложение импульсного сигнала в частичный ряд Фурье.

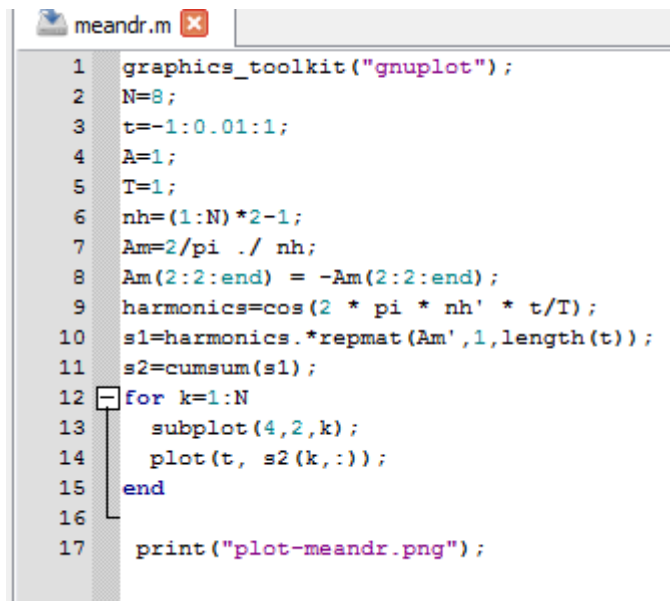
2.1. Построение графиков меандра (рис. 2.1.1 – 2.1.2).

Использовано:

- Формула разложения импульсного сигнала в форме меандра в частичный ряд Фурье для \cos ;
- Цикл для построения отдельных графиков в одном окне;
- Функции `plot` и `subplot`.

Формула разложения импульсного сигнала в форме меандра в частичный ряд Фурье для \cos :

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



```
1 graphics_toolkit("gnuplot");
2 N=8;
3 t=-1:0.01:1;
4 A=1;
5 T=1;
6 nh=(1:N)*2-1;
7 Am=2/pi ./ nh;
8 Am(2:2:end) = -Am(2:2:end);
9 harmonics=cos(2 * pi * nh' * t/T);
10 s1=harmonics.*repmat(Am',1,length(t));
11 s2=cumsum(s1);
12 for k=1:N
13     subplot(4,2,k);
14     plot(t, s2(k,:));
15 end
16
17 print("plot-meandr.png");
```

Рис. 2.1.1. Код построения графиков меандра для \cos .

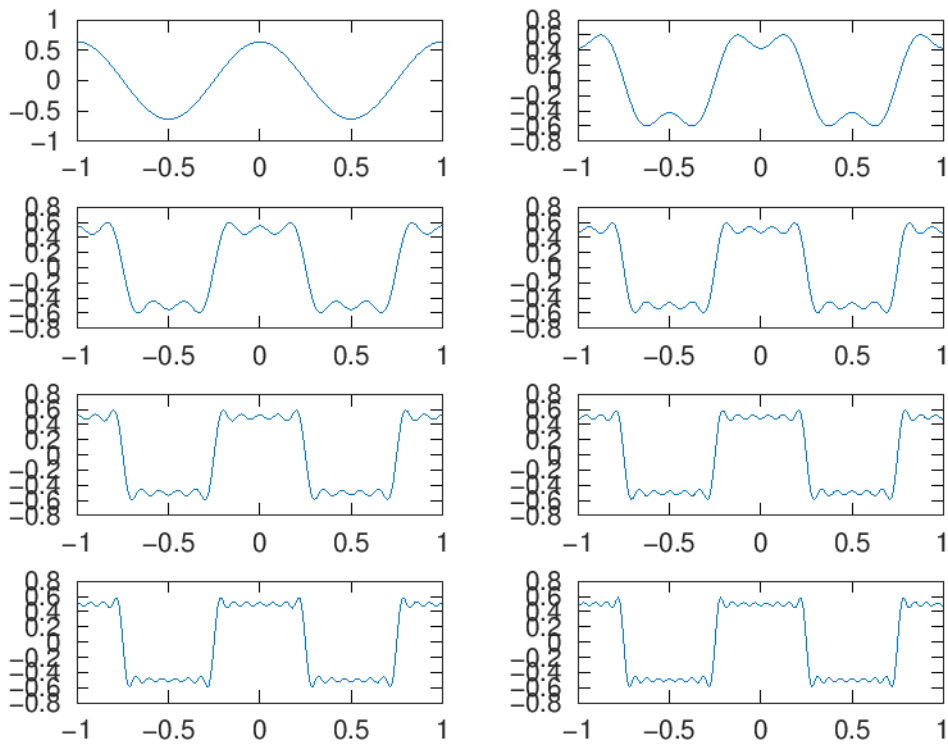


Рис. 2.1.2. Графики меандра для \cos .

2.2. Корректировка для синусов (рис. 2.2.1 – 2.2.2).

Использовано:

- Формула разложения импульсного сигнала в форме меандра в частичный ряд Фурье для \sin ;
- Код из пункта 2.1 .

Формула разложения импульсного сигнала в форме меандра в частичный ряд Фурье для \sin :

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

```

1 graphics_toolkit("gnuplot");
2 N=8;
3 t=-1:0.01:1;
4 A=1;
5 T=1;
6 nh=(1:N)*2-1;
7 Am=2/pi ./ nh;
8 Am(2:2:end) = -Am(2:2:end);
9 harmonics=sin(2 * pi * nh' * t/T);
10 s1=harmonics.*repmat(Am',1,length(t));
11 s2=cumsum(s1);
12 for k=1:N
13     subplot(4,2,k);
14     plot(t, s2(k,:));
15 end
16
17 print("plot-meandr2.png");

```

Рис. 2.2.1. Код построения графиков меандра для \sin .

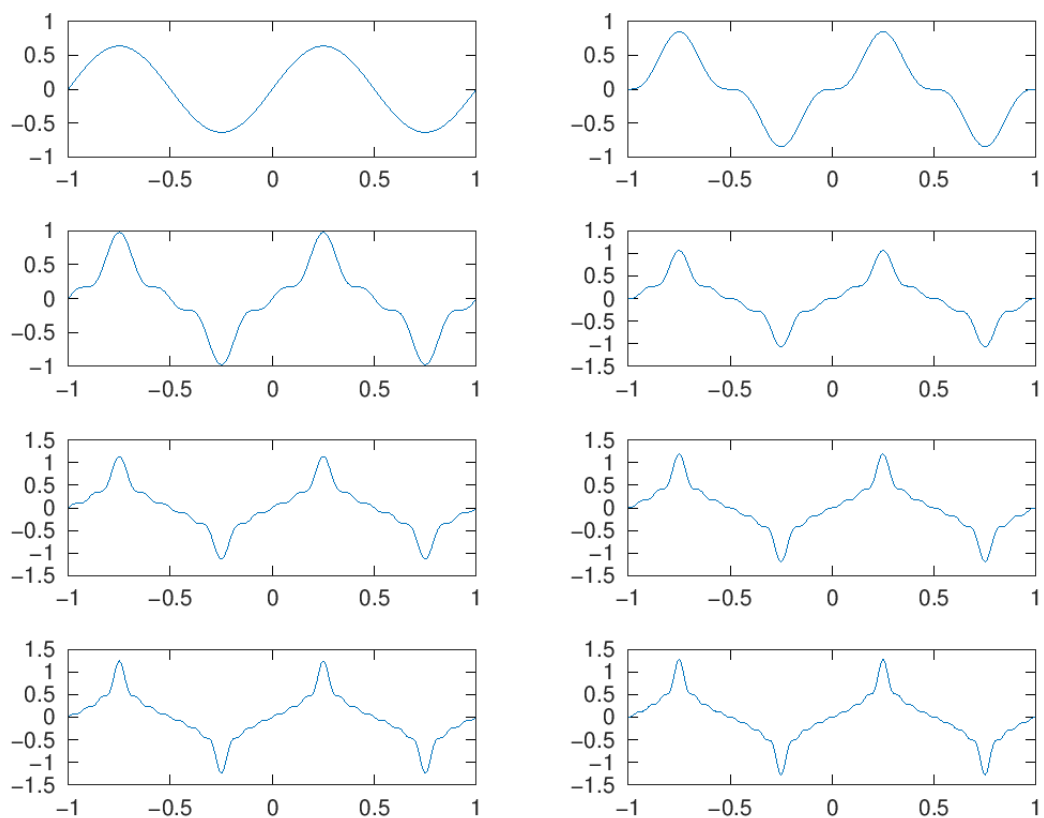


Рис. 2.2.2. Графики меандра для \sin .

3. Определение спектра и параметров сигнала.

3.1. Создание сигнала (рис. 3.1 – 3.3).

Использовано:

- Графический пользовательский интерфейс
- Исходный код сигнала из предоставленного файла ЛР

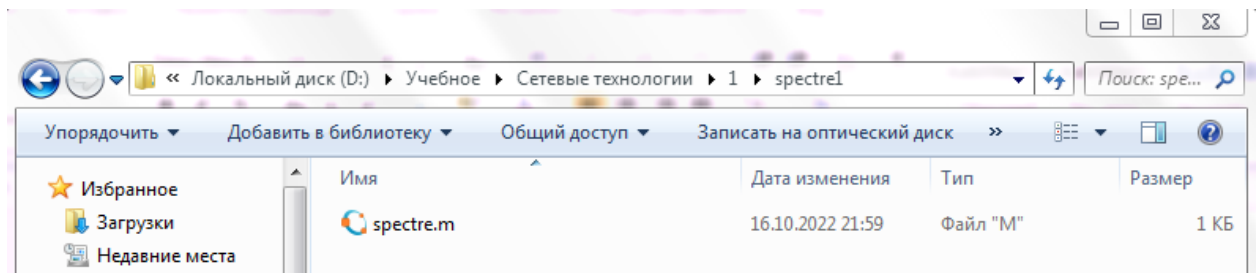


Рис. 3.1. Создание сценария в новом каталоге.

```
spectre.m x
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';
```

Рис. 3.1.2. Код

Результат:

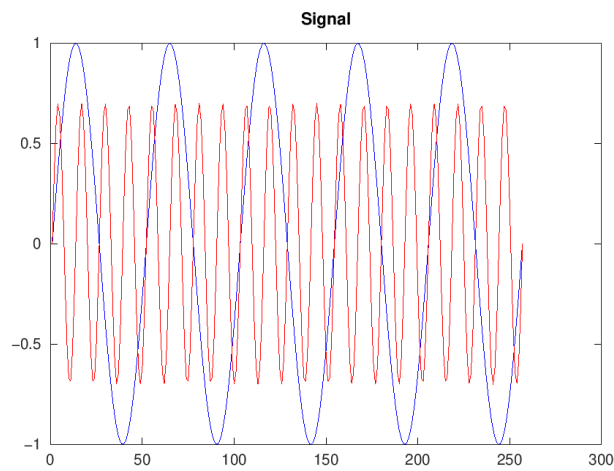


Рис. 3.1.3. График сигнала.

3.2. Получение спектров сигнала (рис. 3.2.1 – 3.2.2).

Использовано:

- Код быстрого преобразования Фурье из файла ЛР.

```
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';
```

Рис. 3.2.1. Код.

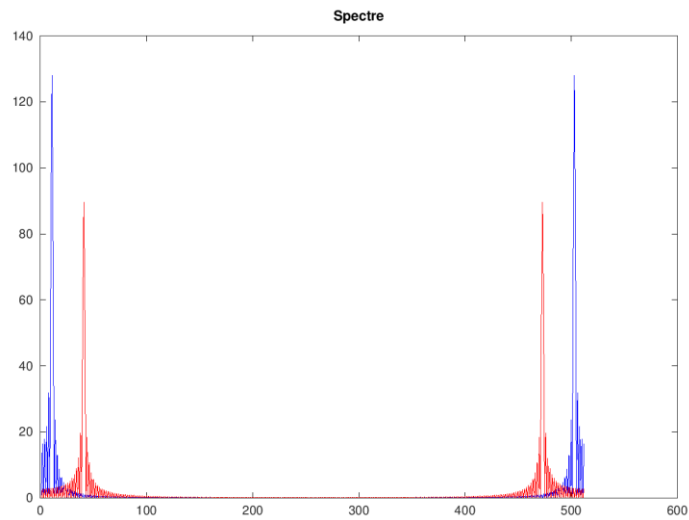


Рис. 3.2.2. График спектра.

3.3. Корректировка графика спектра (рис. 3.3.1 – 3.3.2).

Использовано:

- Отброс дублирующих отрицательных частот
- Учёт суммирования амплитуд на каждом шагу вычисления быстрого преобразования Фурье.

```
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';
```

Рис. 3.3.1. Код.

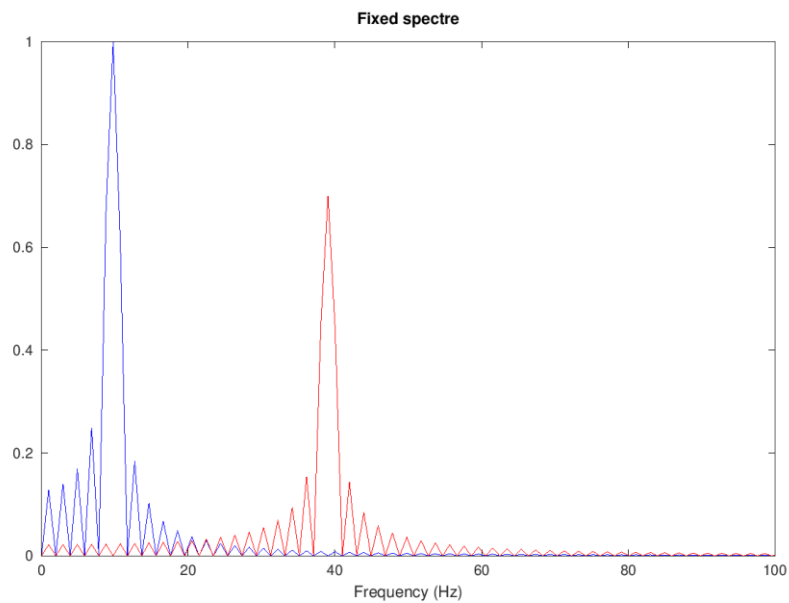


Рис. 3.3.2. Обновлённый график спектра.

3.4. Сумма и спектр суммы рассмотренных сигналов (рис. 3.4.1 – 3.4.3).

Использовано:

- Создание нового сценария в новом каталоге;
- Код из файла ЛР.

```
spectre_sum.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 fd2 = fd/2;
11 t = 0:1./fd:tmax;
12 signal1 = a1*sin(2*pi*t*f1);
13 signal2 = a2*sin(2*pi*t*f2);
14 signal = signal1 + signal2;
15 plot(signal);
16 title('Signal');
17 print 'signal/spectre_sum.png';
18 spectre = fft(signal,fd);
19 f = 1000*(0:fd2)./(2*fd);
20 spectre = 2*sqrt(spectre.*conj(spectre))./fd2;
21 plot(f,spectre(1:fd2+1))
22 xlim([0 100]);
23 title('Spectre');
24 xlabel('Frequency (Hz)');
25 print 'spectre/spectre_sum.png';
```

Рис. 3.4.1. Код.

Результат:

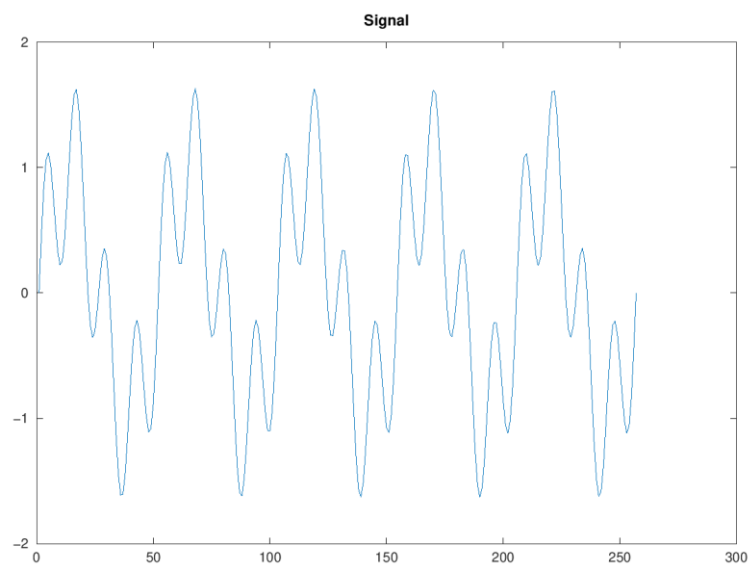


Рис. 3.4.2. Суммарный сигнал

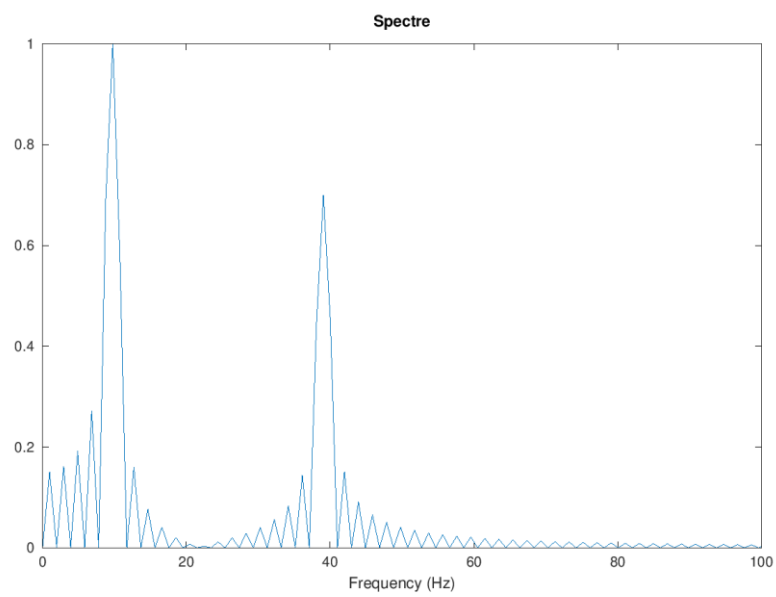
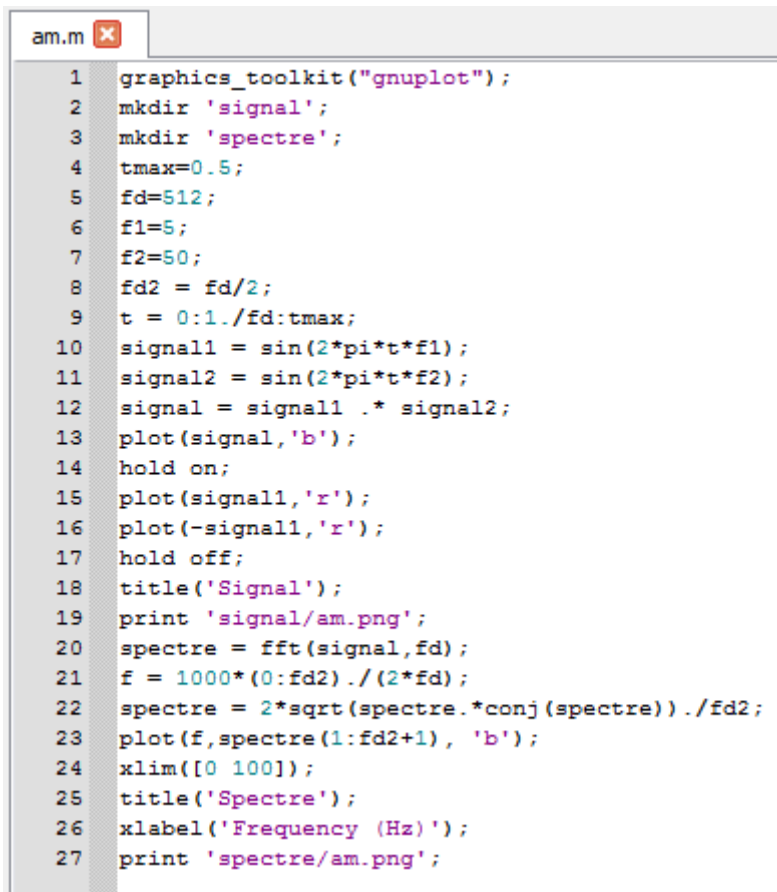


Рис. 3.4.3. Спектр суммарного сигнала.

4. Амплитудная модуляция.

Использовано:

- Код амплитудной модуляции из файла ЛР.



```
1 graphics_toolkit("gnuplot");
2 mkdir 'signal';
3 mkdir 'spectre';
4 tmax=0.5;
5 fd=512;
6 f1=5;
7 f2=50;
8 fd2 = fd/2;
9 t = 0:1./fd:tmax;
10 signal1 = sin(2*pi*t*f1);
11 signal2 = sin(2*pi*t*f2);
12 signal = signal1 .* signal2;
13 plot(signal,'b');
14 hold on;
15 plot(signal1,'r');
16 plot(-signal1,'r');
17 hold off;
18 title('Signal');
19 print 'signal/am.png';
20 spectre = fft(signal,fd);
21 f = 1000*(0:fd2)/(2*fd);
22 spectre = 2*sqrt(spectre.*conj(spectre))./fd2;
23 plot(f,spectre(1:fd2+1), 'b');
24 xlim([0 100]);
25 title('Spectre');
26 xlabel('Frequency (Hz)');
27 print 'spectre/am.png';
```

Рис. 4.1. Код.

Результат.

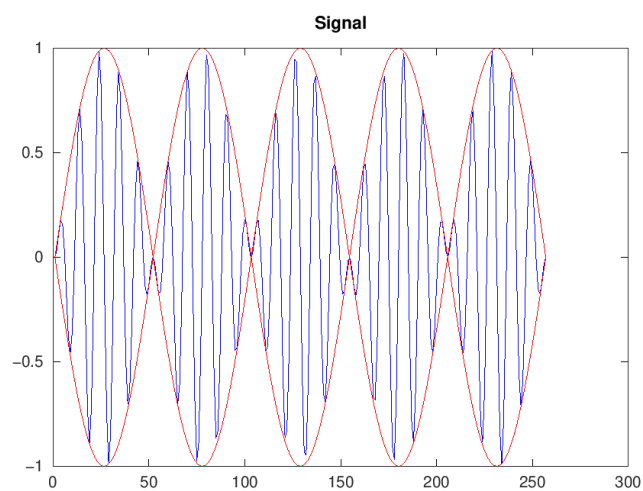


Рис. 4.2. Сигнал и огибающая при амплитудной модуляции

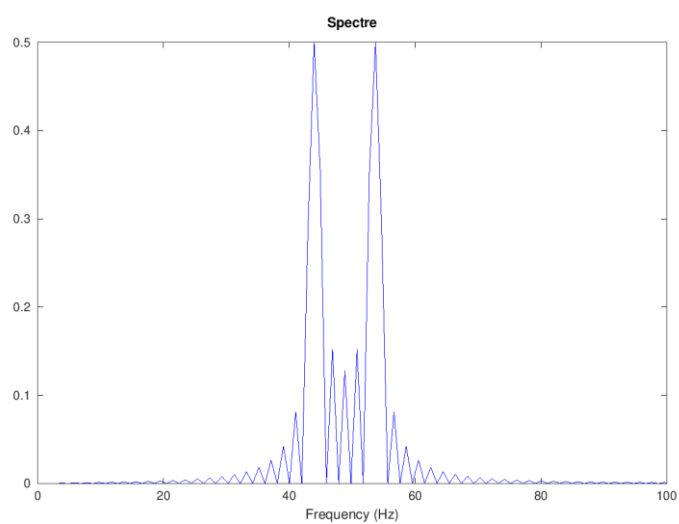
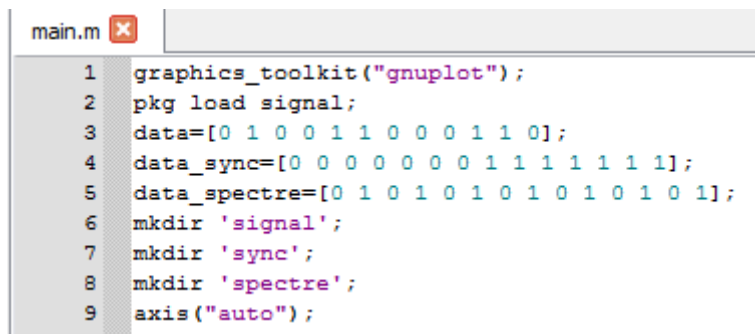


Рис. 4.3. Спектр сигнала при амплитудной модуляции

5. Кодирование сигнала. Исследование самосинхронизации сигнала.

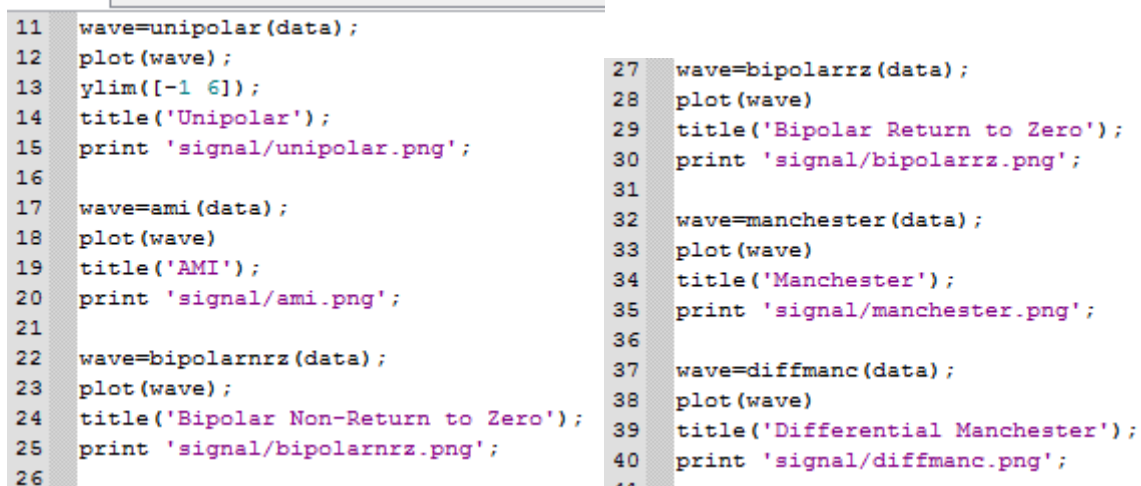
Использовано:

- Пакет signal
- Коды из файла ЛР,



```
1 graphics_toolkit("gnuplot");
2 pkg load signal;
3 data=[0 1 0 0 1 1 0 0 0 1 1 0];
4 data_sync=[0 0 0 0 0 0 0 1 1 1 1 1];
5 data_spectre=[0 1 0 1 0 1 0 1 0 1 0 1];
6 mkdir 'signal';
7 mkdir 'sync';
8 mkdir 'spectre';
9 axis("auto");
```

Рис. 5.1. Код задания входных кодовых последовательностей



```
11 wave=unipolar(data);
12 plot(wave);
13 ylim([-1 6]);
14 title('Unipolar');
15 print 'signal/unipolar.png';
16
17 wave=ami(data);
18 plot(wave);
19 title('AMI');
20 print 'signal/ami.png';
21
22 wave=bipolarnrz(data);
23 plot(wave);
24 title('Bipolar Non-Return to Zero');
25 print 'signal/bipolarnrz.png';
26
27 wave=bipolarrz(data);
28 plot(wave);
29 title('Bipolar Return to Zero');
30 print 'signal/bipolarrz.png';
31
32 wave=manchester(data);
33 plot(wave);
34 title('Manchester');
35 print 'signal/manchester.png';
36
37 wave=diffmanc(data);
38 plot(wave);
39 title('Differential Manchester');
40 print 'signal/diffmanc.png';
41
```

Рис. 5.2. Код вызова функций для построения графиков модуляций кодированных сигналов для кодовой последовательности data.


```

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';

```

Рис. 5.3. Код вызовов функций для построения графиков модуляций кодированных сигналов для кодовой последовательности data_sync.

```

74 wave=unipolar(data_spectre);
75 spectre=calcspectre(wave);
76 title('Unipolar');
77 print 'spectre/unipolar.png';
78
79 wave=ami(data_spectre);
80 spectre=calcspectre(wave);
81 title('AMI');
82 print 'spectre/ami.png';
83
84 wave=bipolarnrz(data_spectre);
85 spectre=calcspectre(wave);
86 title('Bipolar Non-Return to Zero');
87 print 'spectre/bipolarnrz.png';
88
89 wave=bipolarrz(data_spectre);
90 spectre=calcspectre(wave);
91 title('Bipolar Return to Zero');
92 print 'spectre/bipolarrz.png';
93
94 wave=manchester(data_spectre);
95 spectre=calcspectre(wave);
96 title('Manchester');
97 print 'spectre/manchester.png';
98
99 wave=diffmanc(data_spectre);
100 spectre=calcspectre(wave);
101 title('Differential Manchester');
102 print 'spectre/diffmanc.png';

```

Рис. 5.4. Код вызовов функций для графиков спектров.

```

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

```

Рис. 5.5. Код функции, строящей график сигнала по входному битовому потоку.

```

1 function wave=unipolar(data)
2     wave=maptowave(data);
3

```

Рис. 5.6. Код униполярного кодирования.

```

1 function wave=ami(data)
2     am=mod(1:length(data(data==1)),2);
3     am(am==0)=-1;
4     data(data==1)=am;
5     wave=maptowave(data);

```

Рис. 5.7. Код кодирования АМІ.

```

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

```

Рис. 5.8. Кодирование NRZ.

```

1 function wave=bipolarrrz(data)
2     data(data==0)=-1;
3     data=upsample(data,2);
4     wave=maptowave(data);

```

Рис. 5.9. Кодирование RZ.

```

1 function wave=manchester(data)
2     data(data==0)=-1;
3     data=upsample(data,2);
4     data=filter([-1 1],1,data);
5     wave=maptowave(data);
6

```

Рис. 5.10. Манчестерское кодирование.

```

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

```

Рис. 5.11.

Дифференциальное манчестерское кодирование.

```

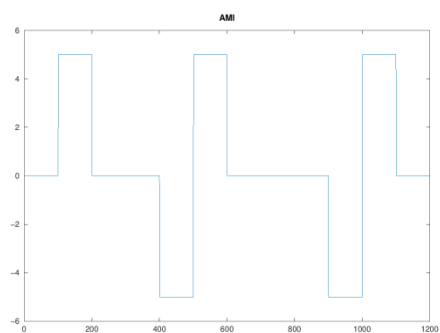
1 function spectre = calcspectre(wave)
2     Fd = 512;
3     Fd2 = Fd/2;
4     Fd3 = Fd/2 + 1;
5
6     X = fft(wave,Fd);
7     spectre = X.*conj(X)/Fd;
8     f = 1000*(0:Fd2)/Fd;
9     plot(f,spectre(1:Fd3));
10    xlabel('Frequency (Hz)');

```

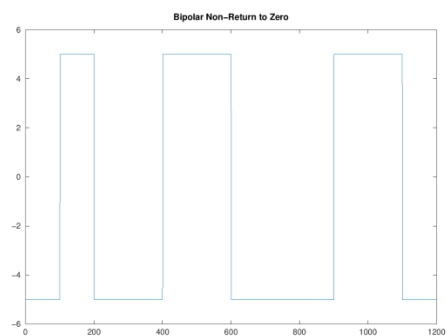
Рис. 5.12. Функция построения графика спектра сигнала.

Результат:

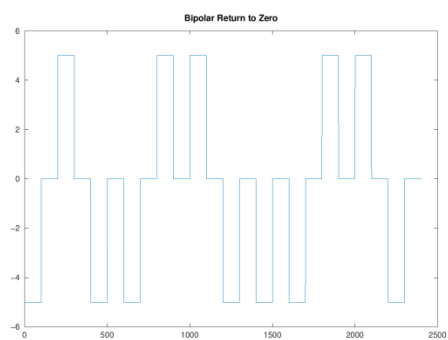
Рис. 5.13. Графики кодированного сигнала:



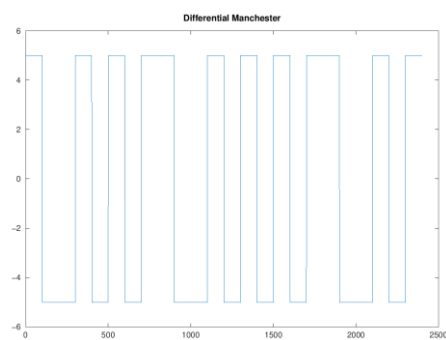
Кодирование AMI



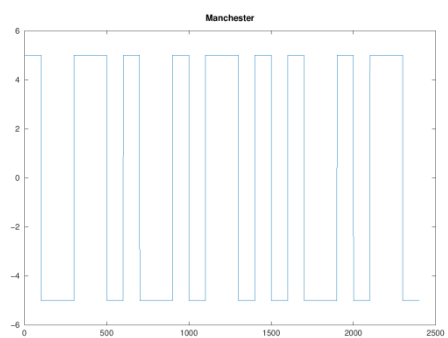
Кодирование NRZ



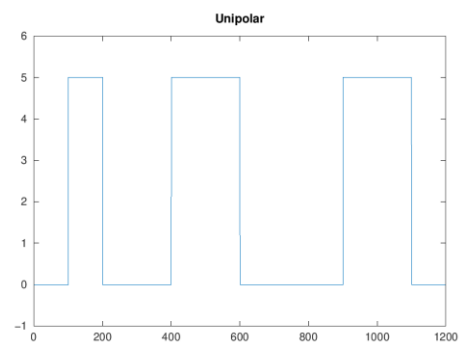
Кодирование RZ



Дифференциальное манчестерское кодирование

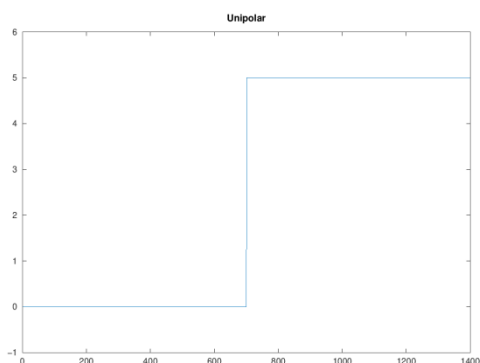


Манчестерское кодирование

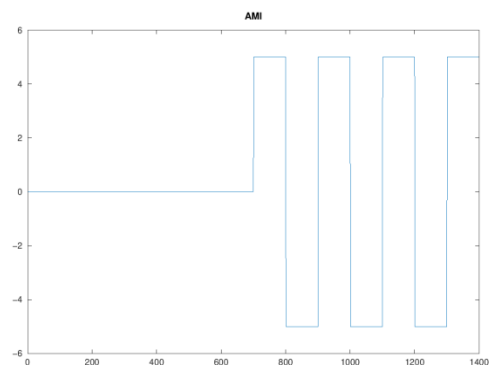


Униполярное кодирование

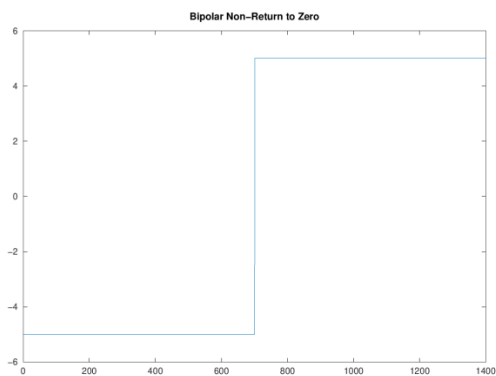
Рис. 5.14. Графики свойств самосинхронизации:



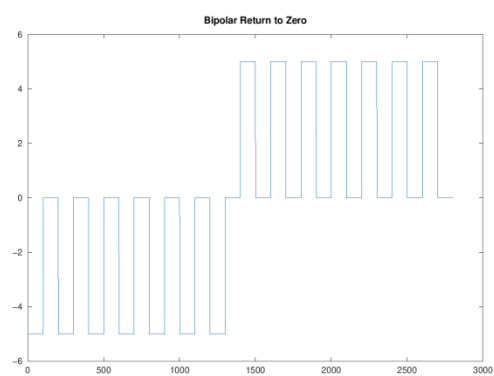
Униполярное кодирование:
нет самосинхронизации



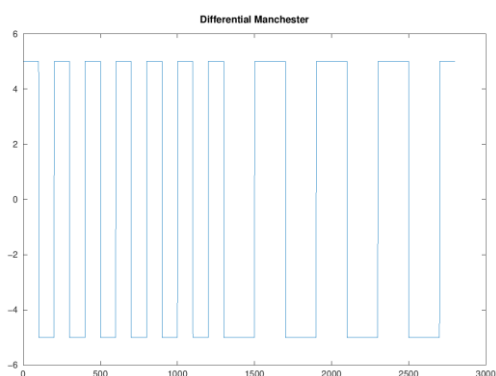
Кодирование AMI:
самосинхронизация при наличии сигнала



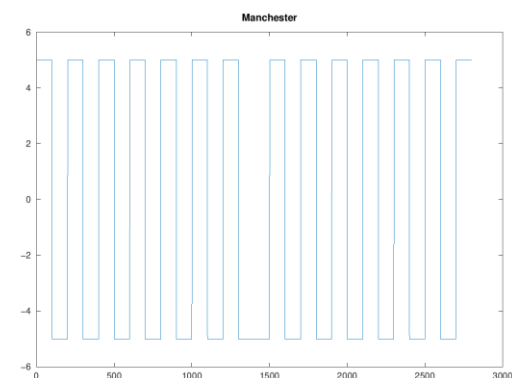
Кодирование NRZ:
нет самосинхронизации



Кодирование RZ:
есть самосинхронизация

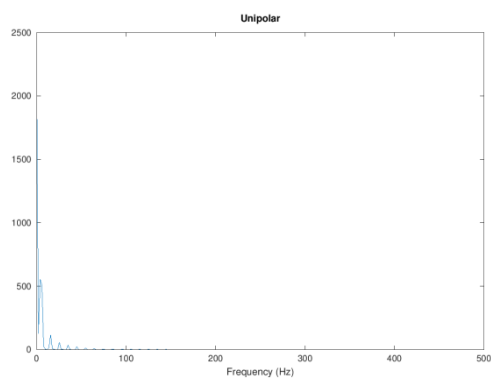


Дифференциальное манчестерское
кодирование: есть самосинхронизация

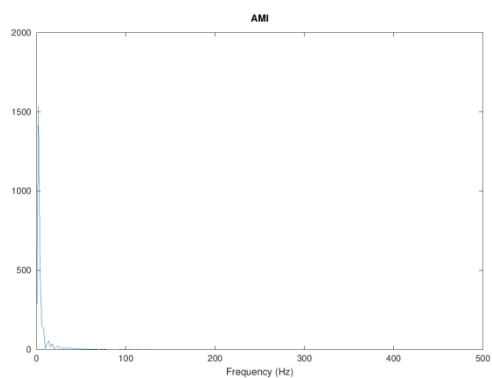


Манчестерское кодирование:
есть самосинхронизация

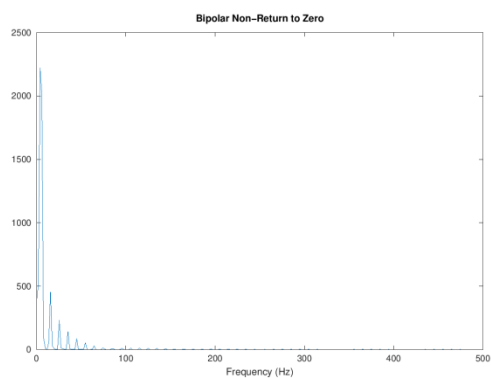
Рис. 5.15. *Графики спектров сигнала:*



Униполярное кодирование



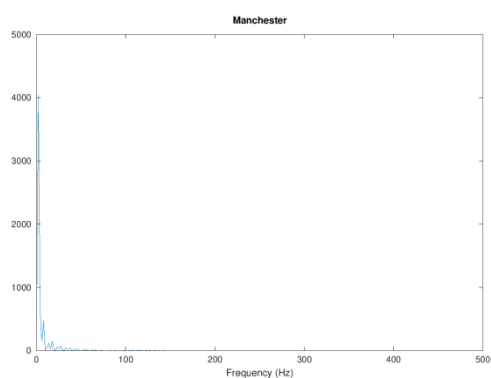
Кодирование AMI



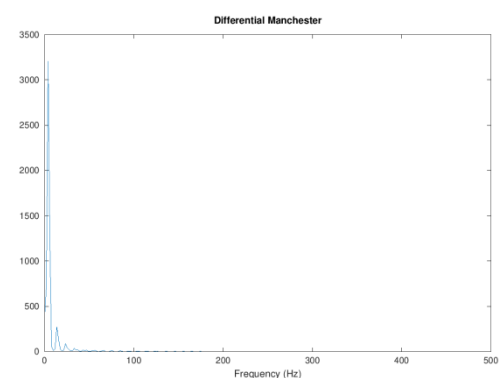
Кодирование NRZ



Кодирование RZ



Манчестерское кодирование



Дифференциальное манчестерское