

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: Саинт-Амур Измаэль

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.

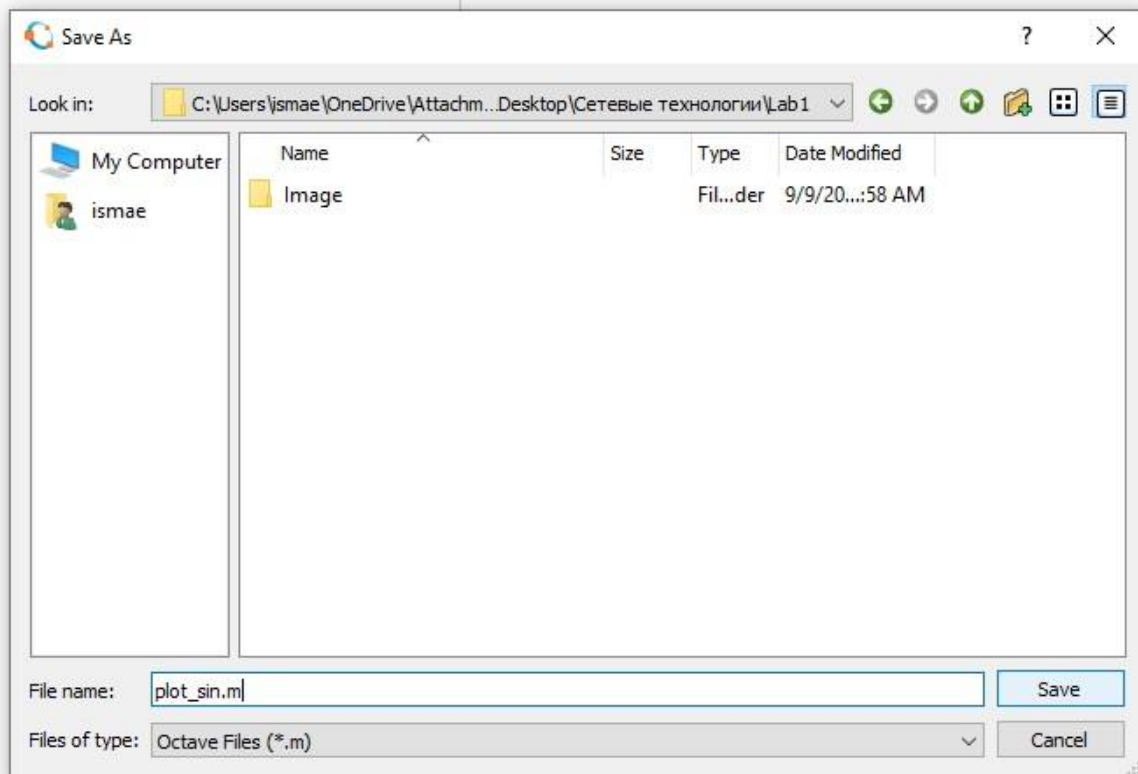
Исследование свойства самосинхронизации сигнала.

1.3.1. Построение графиков в Octave

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

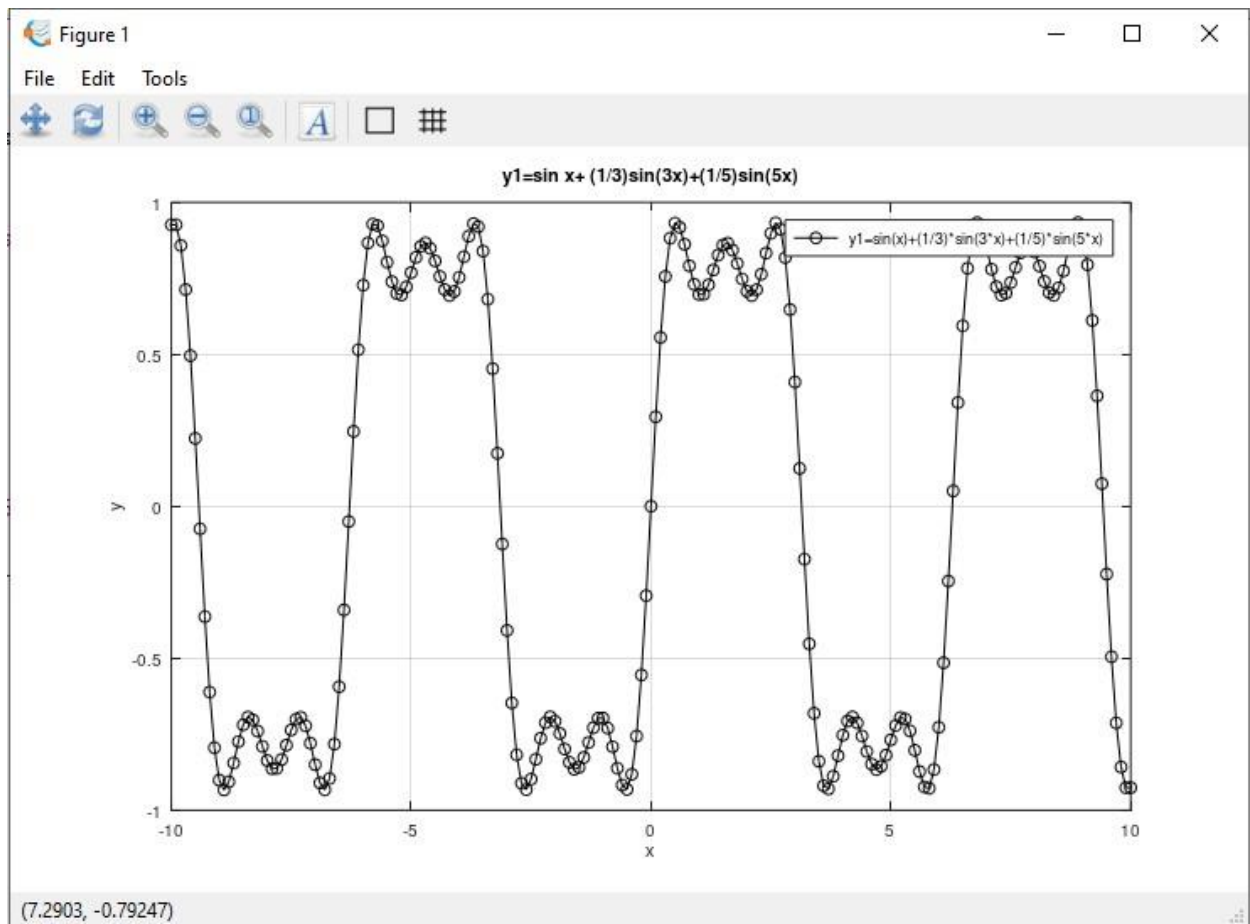
Выполнение работы

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

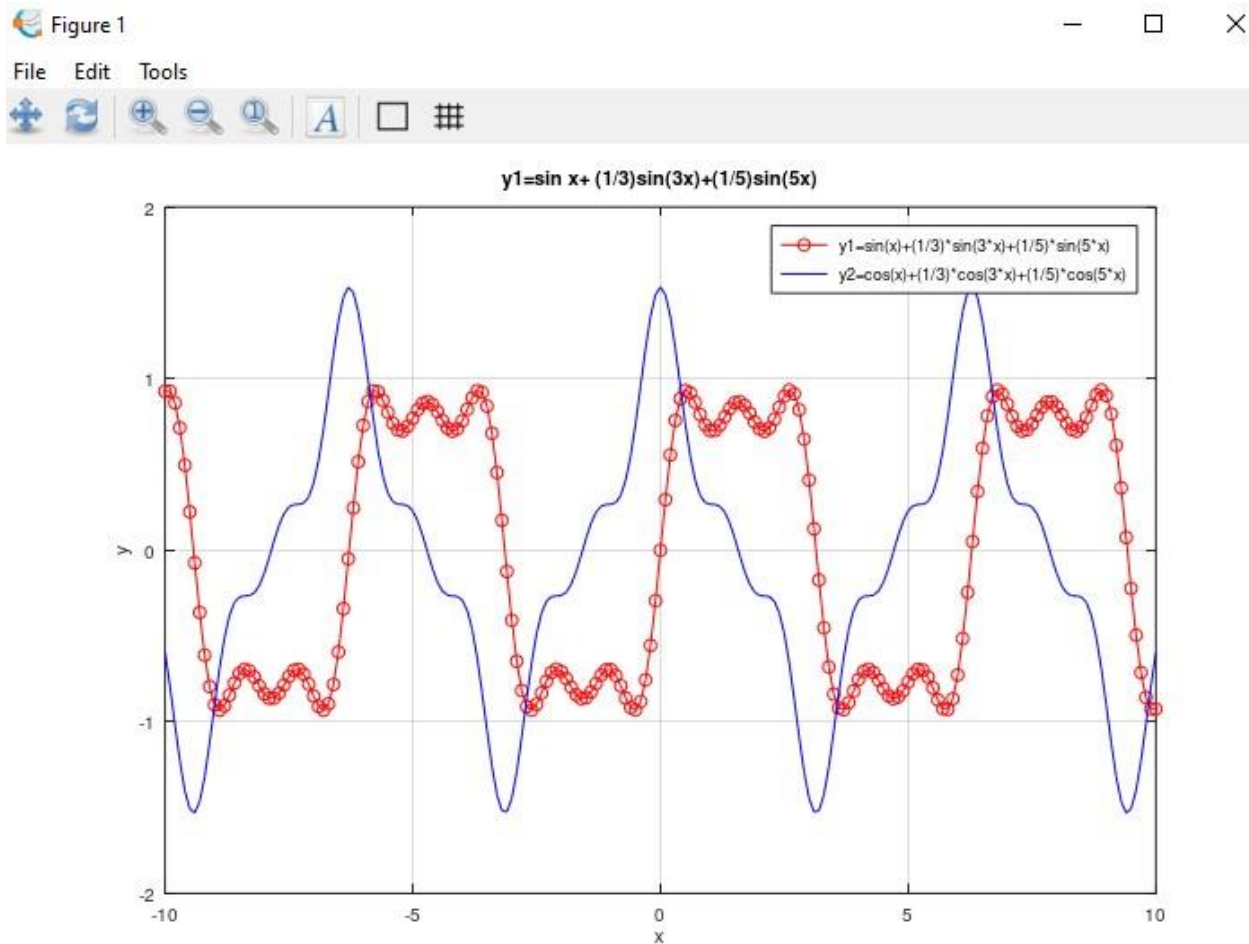


```
plot_sin.m
1 % Формирование массива x:
2 x=-10:0.1:10;
3 % Формирование массива y.
4 y1=sin(x)+1/3*sin(3*x)+1/5*sin(5*x);
5 % Построение графика функции:
6
7 plot(x,y1, "-ok; y1=sin(x)+(1/3)*sin(3*x)+(1/5)*sin(5*x);", "markersize", 4)
8 % Отображение сетки на графике
9 grid on;
10 % Подпись оси X:
11 xlabel('x');
12 % Подпись оси Y:
13 ylabel('y');
14 % Название графика:
15 title('y1=sin x+ (1/3)sin(3x)+(1/5)sin(5x)');
16 % Экспорт рисунка в файл .eps:
17 print ("plot-sin.eps", "-mono", "-FArial:16", "-deps")
18 % Экспорт рисунка в файл .png:
19 print("plot-sin.png");|
```

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



5. Сохраните сценарий под другим названием и измените его так, чтобы на одном графике располагались отличающиеся по типу линий графики функций $y = \cos x + \frac{1}{3} \cos 3x + \frac{1}{5} \cos 5x$



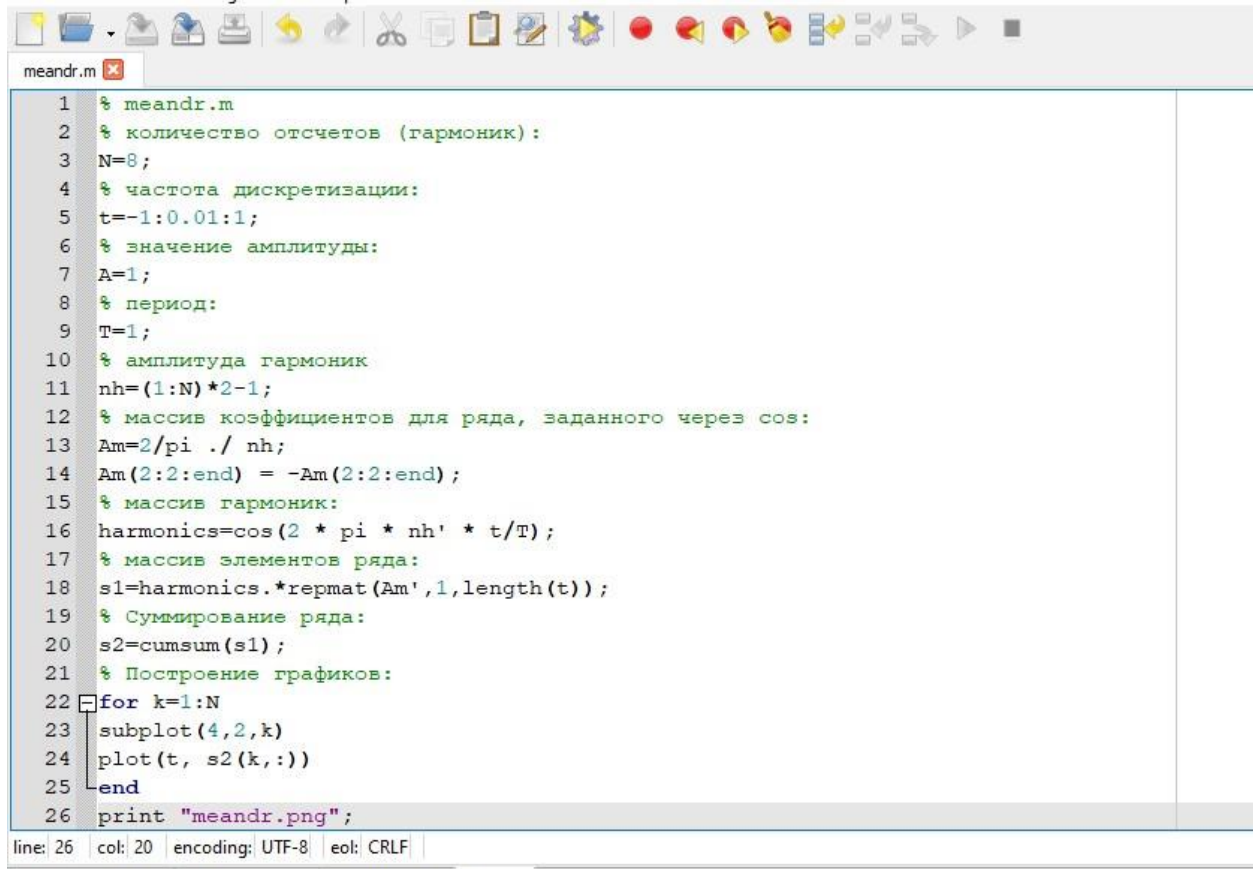
1.3.2. Разложение импульсного сигнала в частичный ряд Фурье

1. Создайте новый сценарий и сохраните его в ваш рабочий каталог с именем, например, meandr.m.
2. В коде созданного сценария задайте начальные значения:
3. Разложение импульсного сигнала в форме меандра в частичный ряд Фурье можно задать формулой

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

или формулой

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



```
1 % meandr.m
2 % количество отсчетов (гармоник):
3 N=8;
4 % частота дискретизации:
5 t=-1:0.01:1;
6 % значение амплитуды:
7 A=1;
8 % период:
9 T=1;
10 % амплитуда гармоник
11 nh=(1:N)*2-1;
12 % массив коэффициентов для ряда, заданного через cos:
13 Am=2/pi ./ nh;
14 Am(2:2:end) = -Am(2:2:end);
15 % массив гармоник:
16 harmonics=cos(2 * pi * nh' * t/T);
17 % массив элементов ряда:
18 s1=harmonics.*repmat(Am',1,length(t));
19 % Суммирование ряда:
20 s2=cumsum(s1);
21 % Построение графиков:
22 for k=1:N
23     subplot(4,2,k)
24     plot(t, s2(k,:))
25 end
26 print "meandr.png";
```

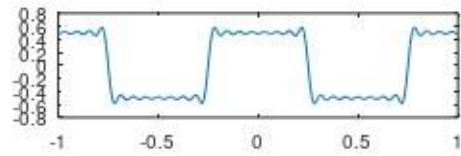
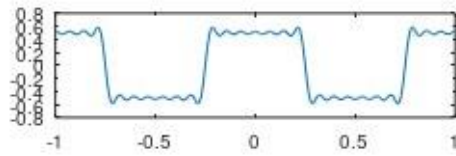
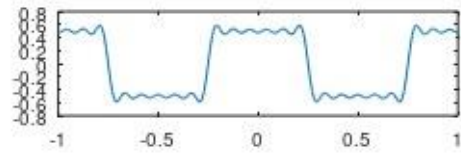
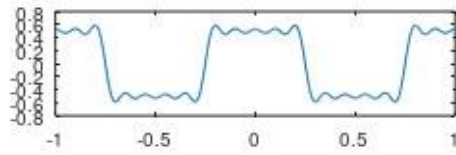
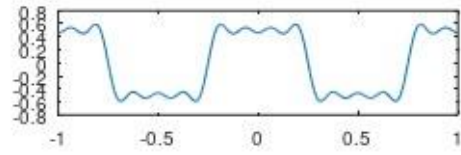
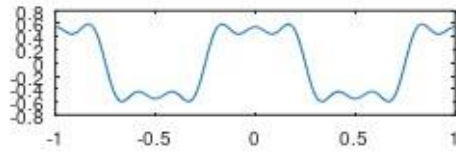
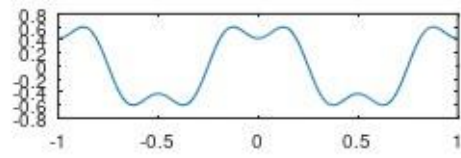
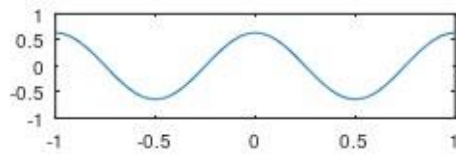
line: 26 col: 20 encoding: UTF-8 eol: CRLF

```
meandr.m
1 % meandr.m
2 % количество отсчетов (гармоник):
3 N=8;
4 % частота дискретизации:
5 t=-1:0.01:1;
6 % значение амплитуды:
7 A=1;
8 % период:
9 T=1;
10 % амплитуда гармоник
11 nh=(1:N)*2-1;
12 % массив коэффициентов для ряда, заданного через cos:
13 Am=2/pi ./ nh;
14 Am(2:2:end) = -Am(2:2:end);
15 % массив гармоник:
16 harmonics=sin(2 * pi * nh' * t/T);
17 % массив элементов ряда:
18 s1=harmonics.*repmat(Am',1,length(t));
19 % Суммирование ряда:
20 s2=cumsum(s1);
21 % Построение графиков:
22 for k=1:N
23     subplot(4,2,k)
24     plot(t, s2(k,:))
25 end
26 print "meandr01.png";
```

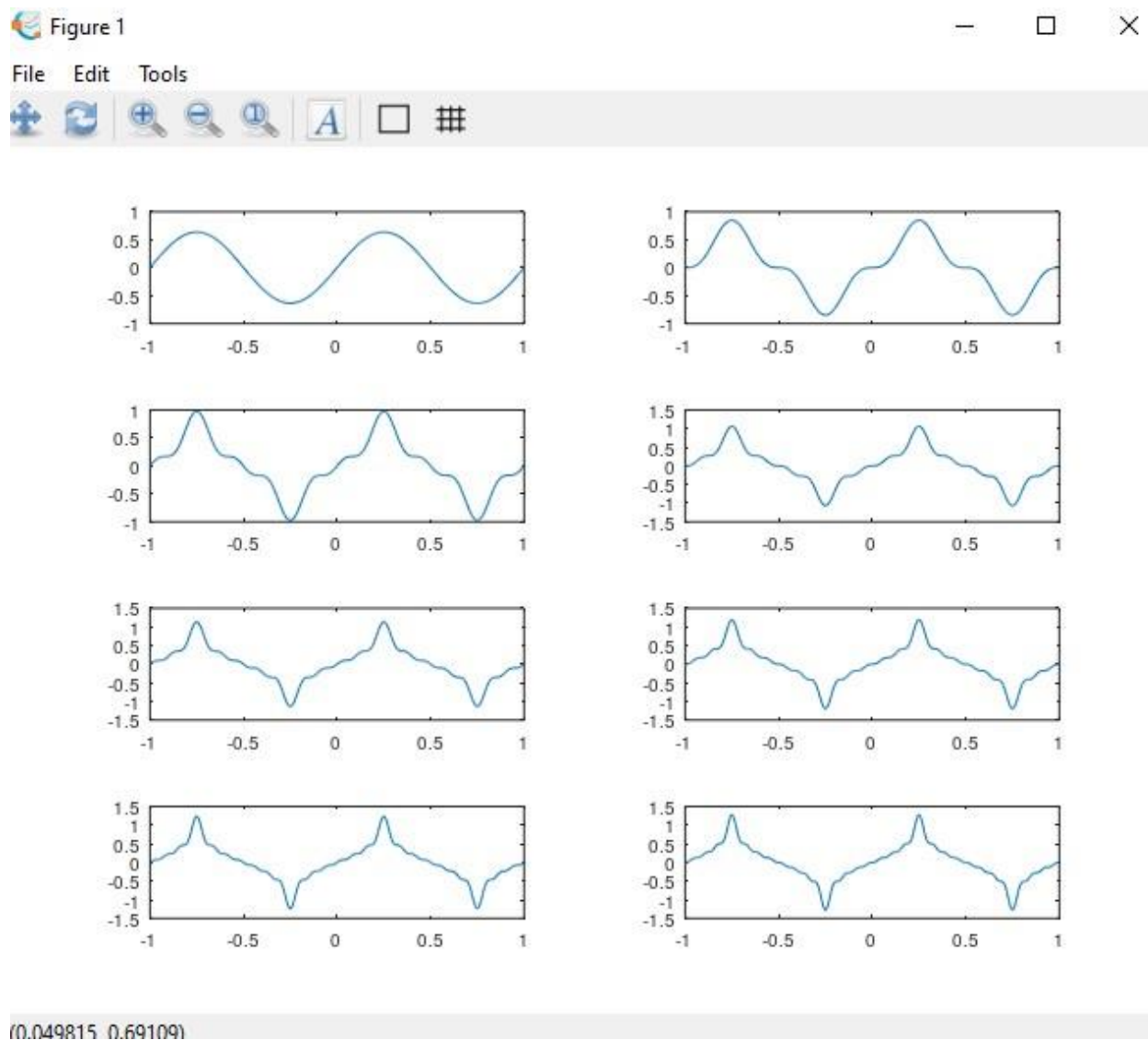
4. Далее для построения в одном окне отдельных графиков меандра с различным количеством гармоник реализуем суммирование ряда с накоплением и воспользуемся функциями `subplot` и `plot` для построения графиков:
5. Экпортируйте полученный график в файл в формате `.png`.
6. Скорректируйте код для реализации меандра через синусы. Получите соответствующие графики.

Figure 1

File Edit Tools

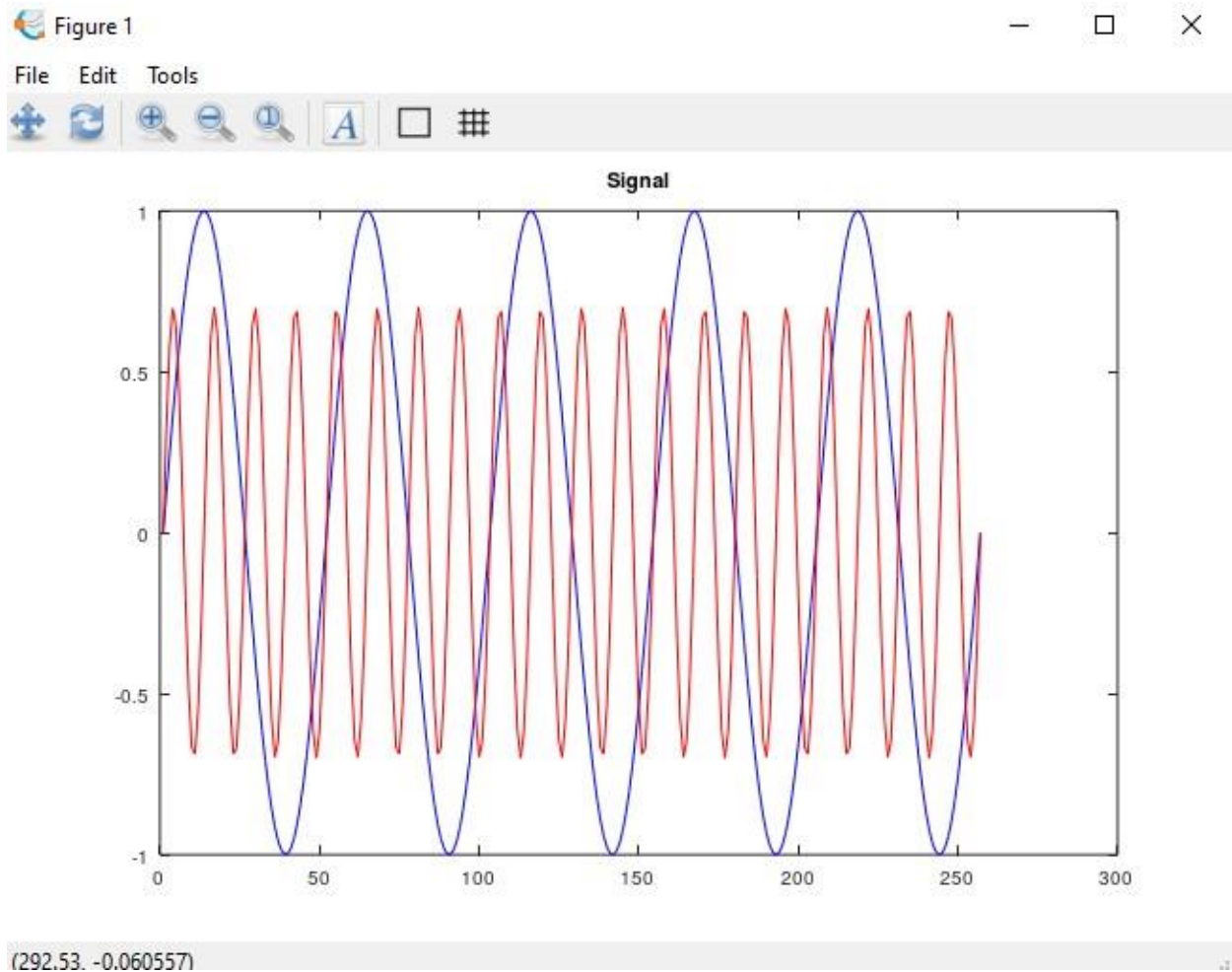


(-0.40812, -0.71386)



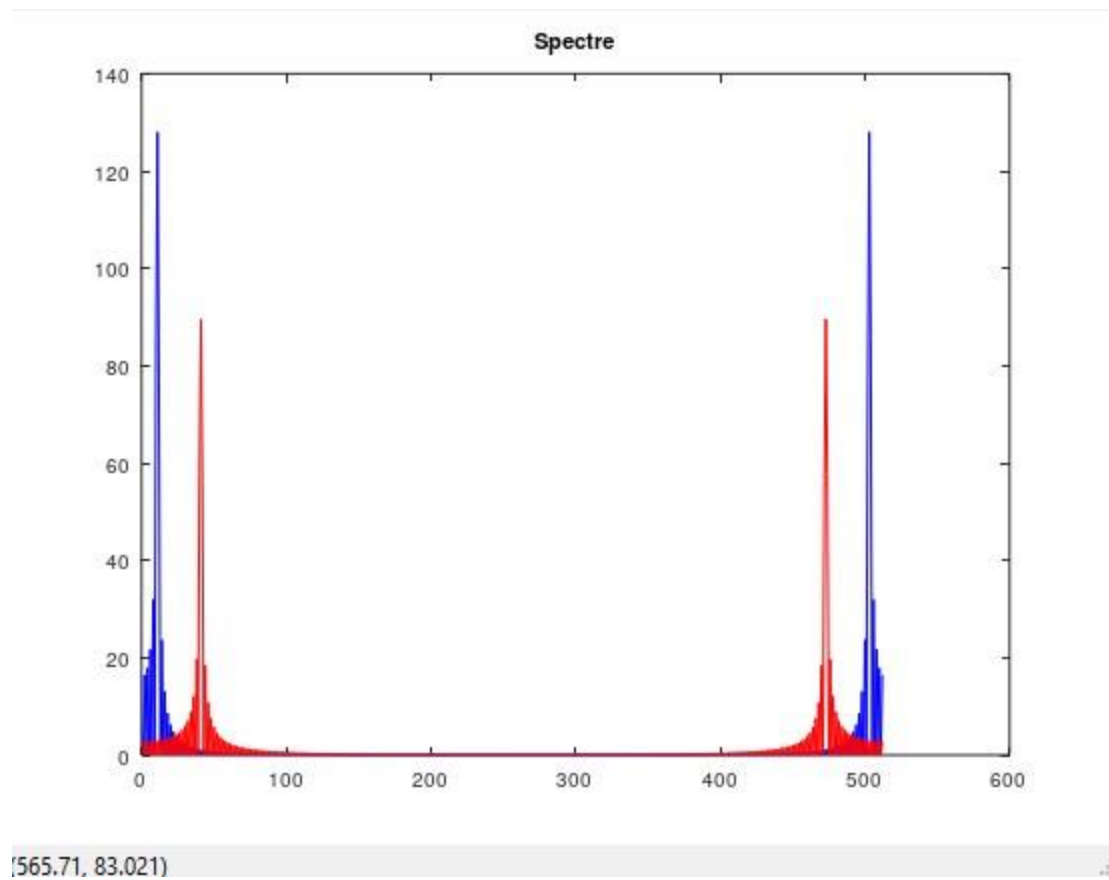
1.3.3. Определение спектра и параметров сигнала

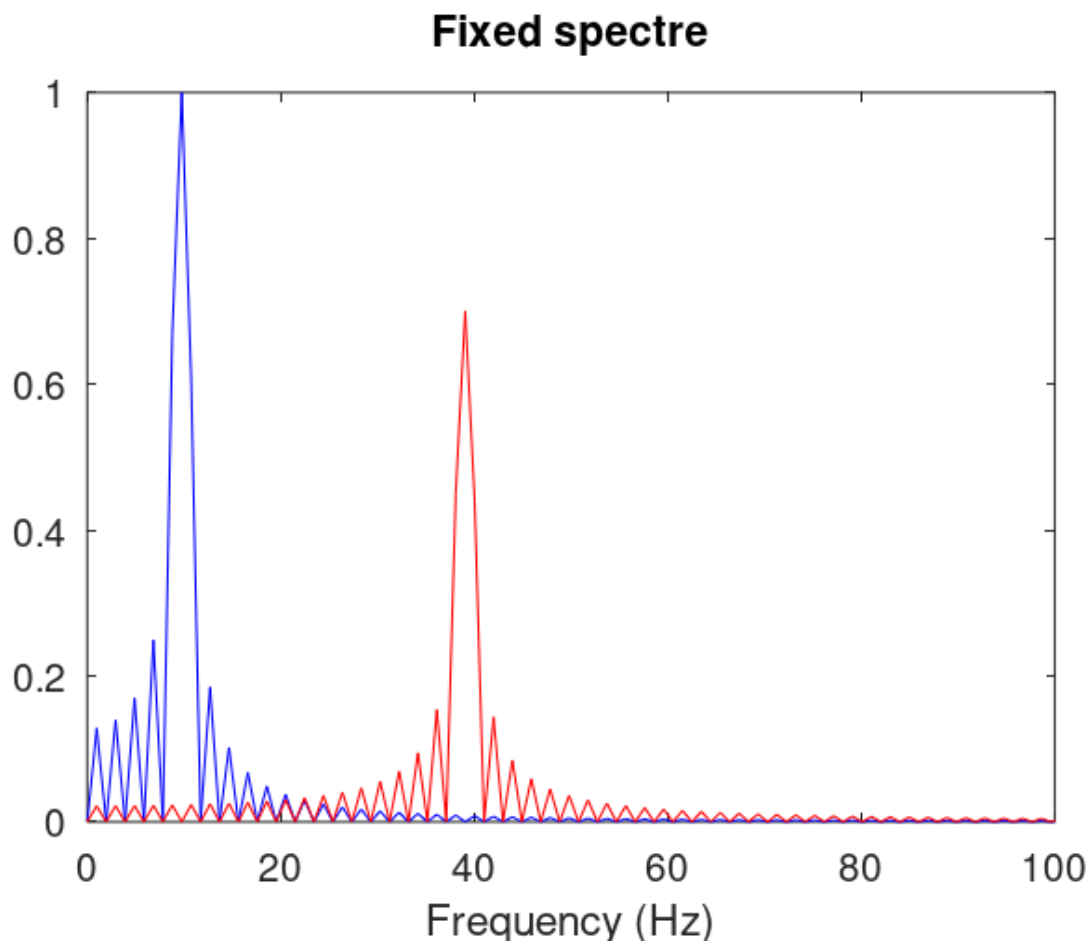
1. В вашем рабочем каталоге создайте каталог `spectre1` и в нём новый сценарий с именем, `spectre.m`.
2. В коде созданного сценария задайте начальные значения:
3. Далее в коде задайте два синусоидальных сигнала разной частоты:
4. Постройте графики сигналов



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 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:





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

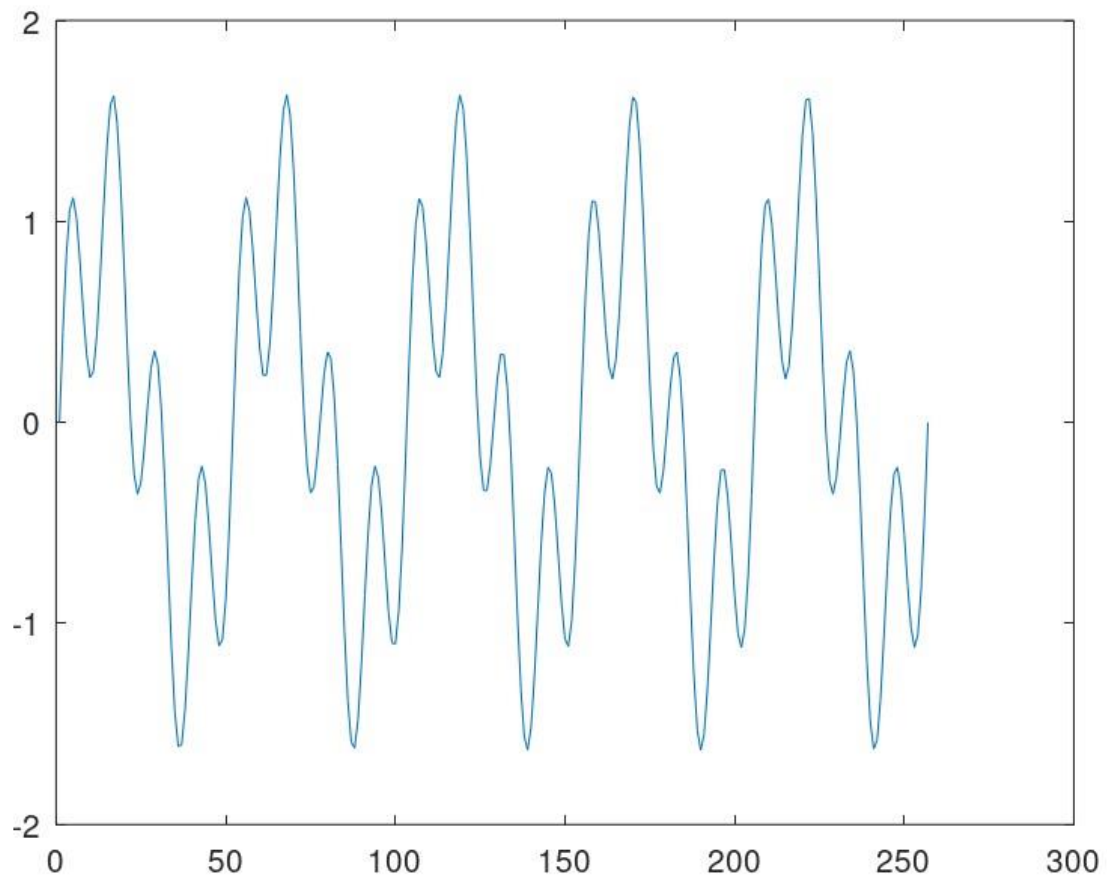
```
spectre.m
1 % spectre1/spectre.m
2 % Создание каталогов signal и spectre для размещения графиков:
3 mkdir 'signal';
4 mkdir 'spectre';
5 % Длина сигнала (с):
6 tmax = 0.5;
7 % Частота дискретизации (Гц) (количество отсчётов):
8 fd = 512;
9 % Частота первого сигнала (Гц):
10 f1 = 10;
11 % Частота второго сигнала (Гц):
12 f2 = 40;
13 % Амплитуда первого сигнала:
14 a1 = 1;
15 % Амплитуда второго сигнала:
16 a2 = 0.7;
17 % Массив отсчётов времени:
18 t = 0:1./fd:tmax;
19 % Спектр сигнала:
20 fd2 = fd/2;
21 % Два сигнала разной частоты:
22 signal1 = a1*sin(2*pi*t*f1);
23 signal2 = a2*sin(2*pi*t*f2);
24 % График 1-го сигнала:
25 plot(signal1,'b');
```

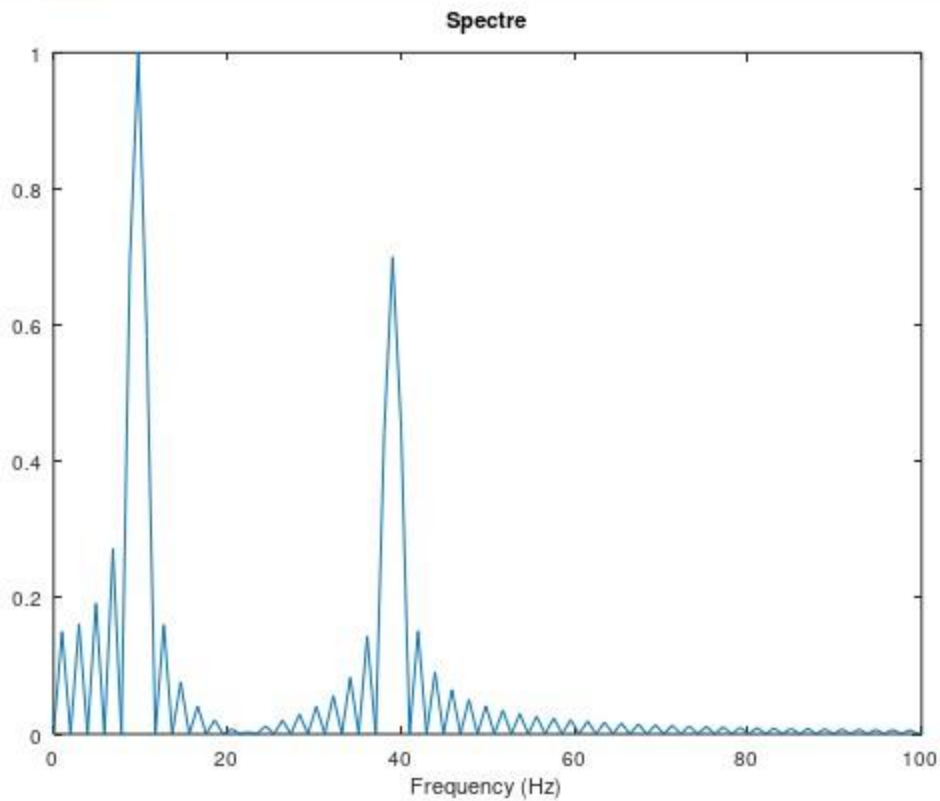
```
spectre.m
25 plot(signal1,'b');
26 % График 2-го сигнала:
27 hold on
28 plot(signal2,'r');
29 hold off
30 title('Signal');
31 print 'signal/spectre.png';
32 % Посчитаем спектр
33 % Амплитуды преобразования Фурье сигнала 1:
34 spectre1 = abs(fft(signal1,fd));
35 % Амплитуды преобразования Фурье сигнала 2:
36 spectre2 = abs(fft(signal2,fd));
37 % Построение графиков спектров сигналов:
38 plot(spectre1,'b');
39 hold on
40 plot(spectre2,'r');
41 hold off
42 title('Spectre');
43 print 'spectre/spectre.png';
44 % Исправление графика спектра
45 % Сетка частот:
46 f = 1000*(0:fd2)/(2*fd);
47 % Нормировка спектров по амплитуде:
48 spectre1 = 2*spectre1/fd2;
49 spectre2 = 2*spectre2/fd2;
```

```
43 print 'spectre/spectre.png';
44 % Исправление графика спектра
45 % Сетка частот:
46 f = 1000*(0:fd2)./(2*fd);
47 % Нормировка спектров по амплитуде:
48 spectre1 = 2*spectre1/fd2;
49 spectre2 = 2*spectre2/fd2;
50 % Построение графиков спектров сигналов:
51 plot(f,spectre1(1:fd2+1),'b');
52 hold on
53 plot(f,spectre2(1:fd2+1),'r');
54 hold off
55 xlim([0 100]);
56 title('Fixed spectre');
57 xlabel('Frequency (Hz)');
58 print 'spectre/spectre_fix.png';
```

line: 58 | col: 33 | encoding: UTF-8 | eol: CRLF

Signal





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:

```
am.m
1 % modulation/am.m
2 % Создание каталогов signal и spectre для размещения графиков:
3 mkdir 'signal';
4 mkdir 'spectre';
5 % Модуляция синусоид с частотами 50 и 5
6 % Длина сигнала (с)
7 tmax = 0.5;
8 % Частота дискретизации (Гц) (количество отсчётов)
9 fd = 512;
10 % Частота сигнала (Гц)
11 f1 = 5;
12 % Частота несущей (Гц)
13 f2 = 50;
14 % Спектр сигнала
15 fd2 = fd/2;
16 % Построение графиков двух сигналов (синусоиды)
17 % разной частоты
18 % Массив отсчётов времени:
19 t = 0:1./fd:tmax;
20 signal1 = sin(2*pi*t*f1);
21 signal2 = sin(2*pi*t*f2);
22 signal = signal1 .* signal2;
23 plot(signal, 'b');
24 hold on
25 % Построение огибающей:
26 plot(signal1, 'r');
```

line: 2 col: 54 encoding: UTF-8 eol: CRLF

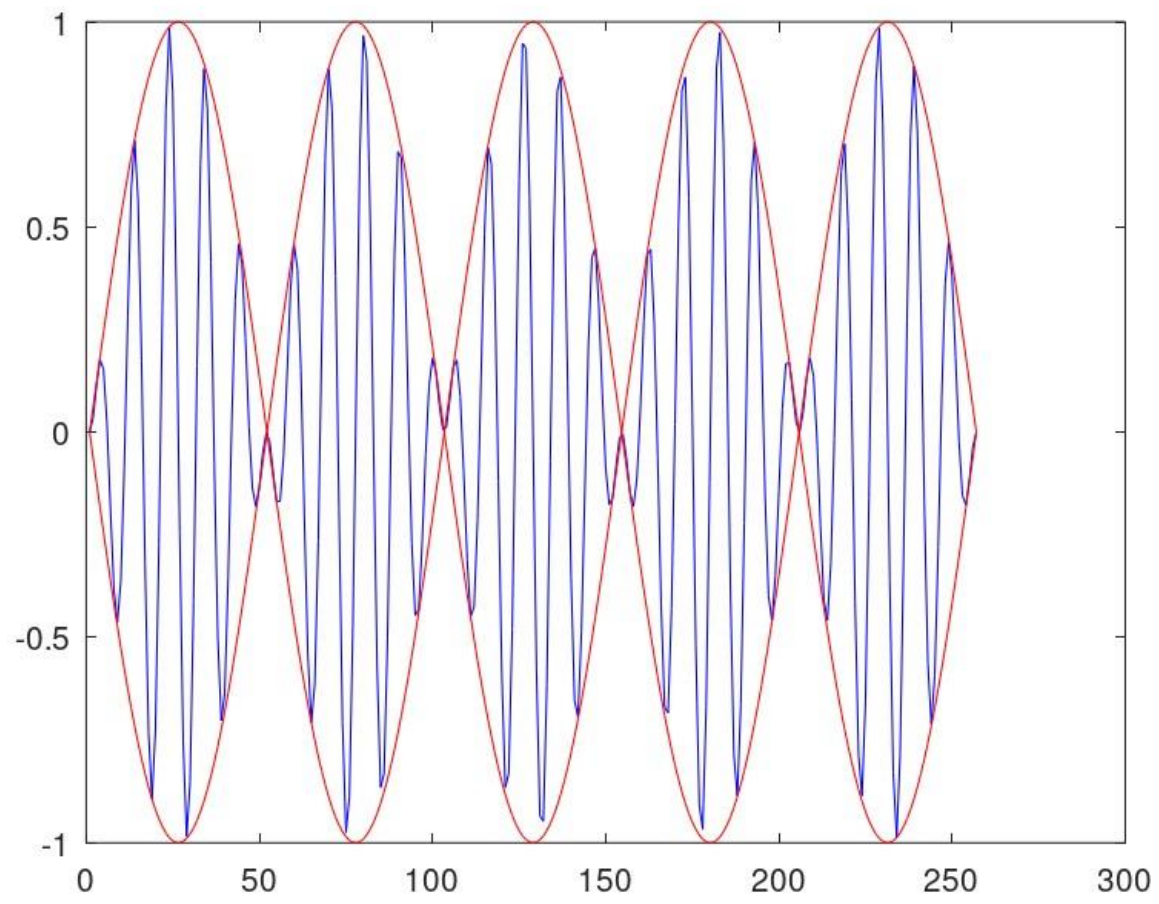
Command Window Documentation Variable Editor Filter

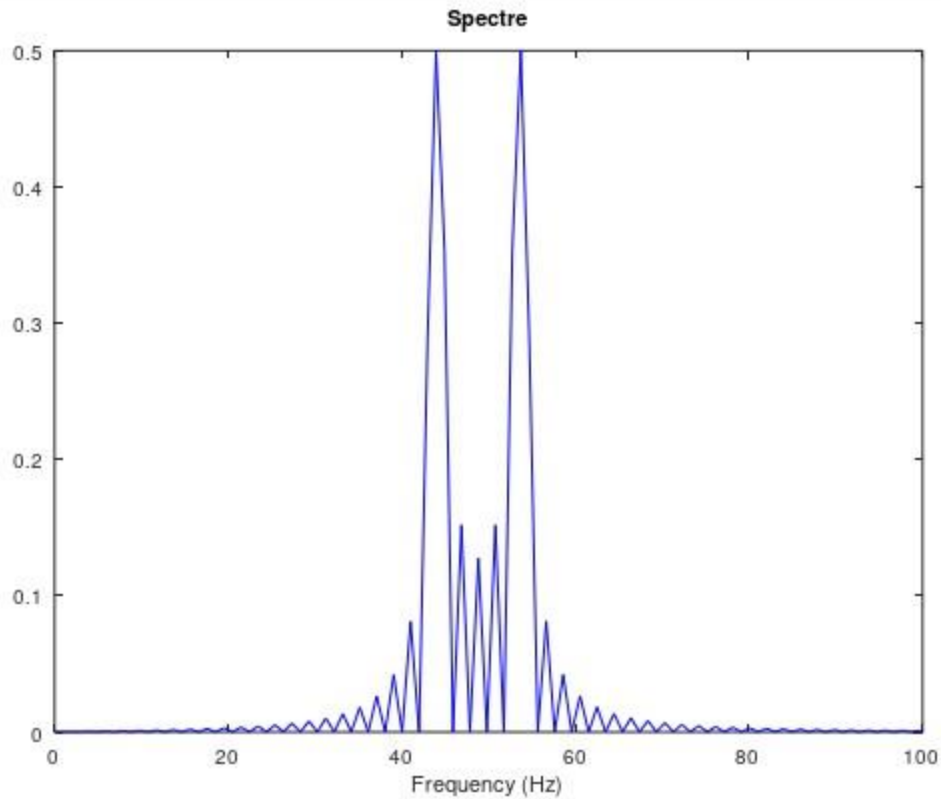
```
am.m
19 t = 0:1./fd:tmax;
20 signal1 = sin(2*pi*t*f1);
21 signal2 = sin(2*pi*t*f2);
22 signal = signal1 .* signal2;
23 plot(signal, 'b');
24 hold on
25 % Построение огибающей:
26 plot(signal1, 'r');
27 plot(-signal1, 'r');
28 hold off
29 title('Signal');
30 print 'signal/am.png';
31 % Расчет спектра:
32 % Амплитуды преобразования Фурье-сигнала:
33 spectre = fft(signal,fd);
34 % Сетка частот:
35 f = 1000*(0:fd2)/(2*fd);
36 % Нормировка спектра по амплитуде:
37 spectre = 2*sqrt(spectre.*conj(spectre))./fd2;
38 % Построение спектра:
39 plot(f,spectre(1:fd2+1), 'b')
40 xlim([0 100]);
41 title('Spectre');
42 xlabel('Frequency (Hz)');
43 print 'spectre/am.png';
```

line: 2 col: 54 encoding: UTF-8 eol: CRLF

В результате получаем, что спектр произведения представляет собой свёртку спектров

Signal





1.3.5. Кодирование сигнала. Исследование свойства самосинхронизации сигнала

1. В вашем рабочем каталоге создайте каталог `coding` и в нём файлы `main.m` `maptowave.m`, `unipolar.m`, `ami.m`, `bipolarnrz.m`, `bipolarrz.m`, `manchester.m`, `diffmanc.m`, `calcspectre.m`.
2. В окне интерпретатора команд проверьте, установлен ли у вас пакет расширений `signal`:
`>> pkg list`

```

>> pkg list
warning: load_path: C:\Users\ismae\OneDrive\Attachments\Desktop\Сетевые технологии\L
warning: load_path: C:\Users\ismae\OneDrive\Attachments\Desktop\Сетевые технологии\L
Package Name      | Version | Installation directory
-----
audio             | 2.0.5   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
biosig            | 2.4.2   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
communications    | 1.2.4   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
control           | 3.4.0   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
data-smoothing    | 1.3.0   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
database          | 2.4.4   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
dataframe         | 1.2.0   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
dicom             | 0.5.0   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
financial         | 0.5.3   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
fits             | 1.0.7   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
fuzzy-logic-toolkit | 0.4.6   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
ga               | 0.10.3  | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
general          | 2.1.2   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
generate_html     | 0.3.3   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
geometry          | 4.0.0   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
gsl              | 2.1.1   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
image            | 2.14.0  | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
instrument-control | 0.8.0   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
interval         | 3.2.1   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
io               | 2.6.4   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
linear-algebra    | 2.2.3   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
lssa             | 0.1.4   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
ltfat            | 2.3.1   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
mapping          | 1.4.2   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
matgeom          | 1.2.3   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
miscellaneous     | 1.3.0   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
nan              | 3.7.0   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
netcdf           | 1.0.14  | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
nurbs            | 1.4.3   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
ocs              | 0.1.5   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
octproj          | 2.0.1   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
optim            | 1.6.2   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
optiminterp      | 0.3.7   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
quaternion       | 2.4.0   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh
queueing         | 1.2.7   | C:\Program Files\GNU Octave\Octave-7.2.0\mingw64\sh

```

<
Command Window
Documentation
Variable Editor
Editor

3. В файле main.m подключите пакет signal и задайте входные кодовые последовательности:


```
main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m x bipolarrrz.m x manchester.m x diffmanc.m x calcspectre.m x
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 1];
8 % Входная кодовая последовательность для построения спектра сигнала:
9 data_spectre=[0 1 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 % Униполярное кодирование
16 wave=unipolar(data);
17 plot(wave);
18 ylim([-1 6]);
19 title('Unipolar');
20 print 'signal/unipolar.png';
21 % Кодирование ami
22 wave=ami(data);
23 plot(wave);
24 title('AMI');
25 print 'signal/ami.png';
26 % Кодирование NRZ
line: 50 col: 1 encoding: UTF-8 eol: CRLF
Command Window Documentation Variable Editor Editor
```

```
main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m x bipolarrrz.m x manchester.m x diffmanc.m x calcspectre.m x
26 % Кодирование NRZ
27 wave=bipolarnrz(data);
28 plot(wave);
29 title('Bipolar Non-Return to Zero');
30 print 'signal/bipolarnrz.png';
31 % Кодирование RZ
32 wave=bipolarrrz(data);
33 plot(wave);
34 title('Bipolar Return to Zero');
35 print 'signal/bipolarrrz.png';
36 % Манчестерское кодирование
37 wave=manchester(data);
38 plot(wave);
39 title('Manchester');
40 print 'signal/manchester.png';
41 % Дифференциальное манчестерское кодирование
42 wave=diffmanc(data);
43 plot(wave);
44 title('Differential Manchester');
45 print 'signal/diffmanc.png';
46 % Дифференциальное манчестерское кодирование
47 wave=diffmanc(data);
48 plot(wave);
49 title('Differential Manchester');
50 print 'signal/diffmanc.png';
51 % Униполярное кодирование
line: 50 col: 1 encoding: UTF-8 eol: CRLF
```

```
main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m x bipolarrrz.m x manchester.m x diffmanc.m x calcspectre.m x
50 print 'signal/diffmanc.png';
51 % Униполярное кодирование
52 wave=unipolar(data_sync);
53 plot(wave);
54 ylim([-1 6]);
55 title('Unipolar');
56 print 'sync/unipolar.png';
57 % Кодирование AMI
58 wave=ami(data_sync);
59 plot(wave)
60 title('AMI');
61 print 'sync/ami.png';
62 % Кодирование NRZ
63 wave=bipolarnrz(data_sync);
64 plot(wave);
65 title('Bipolar Non-Return to Zero');
66 print 'sync/bipolarnrz.png';
67 % Кодирование RZ
68 wave=bipolarrrz(data_sync);
69 plot(wave)
70 title('Bipolar Return to Zero');
71 print 'sync/bipolarrrz.png';
72 % Манчестерское кодирование
73 wave=manchester(data_sync);
74 plot(wave)
75 title('Manchester');
```

line: 50 col: 1 encoding: UTF-8 eol: CRLF

main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m x bipolarrz.m x manchester.n

```
76 print 'sync/manchester.png';
77 % Дифференциальное манчестерское кодирование
78 wave=diffmanc(data_sync);
79 plot(wave)
80 title('Differential Manchester');
81 print 'sync/diffmanc.png';
82 % Униполярное кодирование:
83 wave=unipolar(data_spectre);
84 spectre=calcspectre(wave);
85 title('Unipolar');
86 print 'spectre/unipolar.png';
87 % Кодирование AMI:
88 wave=ami(data_spectre);
89 spectre=calcspectre(wave);
90 title('AMI');
91 print 'spectre/ami.png';
92 % Кодирование NRZ:
93 wave=bipolarnrz(data_spectre);
94 spectre=calcspectre(wave);
95 title('Bipolar Non-Return to Zero');
96 print 'spectre/bipolarnrz.png';
97 % Кодирование RZ:
98 wave=bipolarrz(data_spectre);
99 spectre=calcspectre(wave);
100 title('Bipolar Return to Zero');
101 print 'spectre/bipolarrz.png';
```

line: 50 col: 1 encoding: UTF-8 eol: CRLF

Command Window Documentation Variable Editor Editor

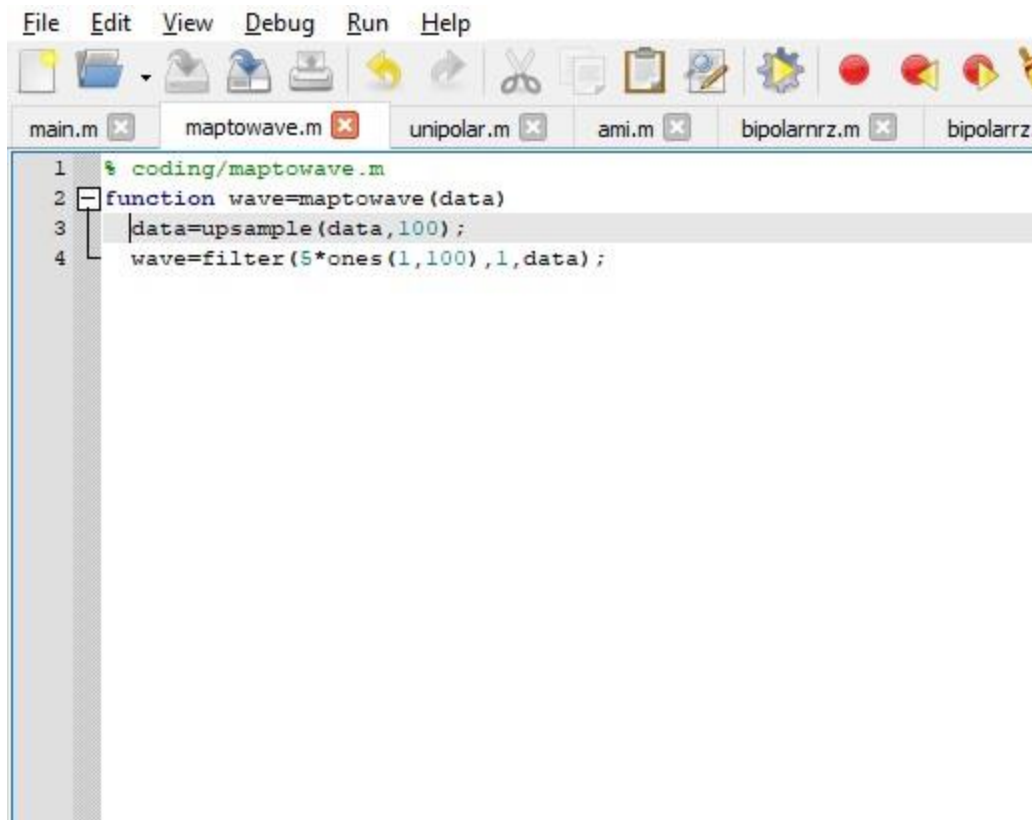

```

93 wave=bipolarnrz(data_spectre);
94 spectre=calcspectre(wave);
95 title('Bipolar Non-Return to Zero');
96 print 'spectre/bipolarnrz.png';
97 % Кодирование RZ:
98 wave=bipolarrz(data_spectre);
99 spectre=calcspectre(wave);
100 title('Bipolar Return to Zero');
101 print 'spectre/bipolarrz.png';
102 % Манчестерское кодирование:
103 wave=manchester(data_spectre);
104 spectre=calcspectre(wave);
105 title('Manchester');
106 print 'spectre/manchester.png';
107 % Дифференциальное манчестерское кодирование:
108 wave=diffmanc(data_spectre);
109 spectre=calcspectre(wave);
110 title('Differential Manchester');
111 print 'spectre/diffmanc.png';

```

line: 50 col: 1 encoding: UTF-8 eol: CRLF

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



The screenshot shows a MATLAB editor window with the following menu bar: File, Edit, View, Debug, Run, Help. The toolbar includes icons for file operations (new, open, save, print, copy, paste, delete), editing (undo, redo, cut, find, replace), and execution (run, stop, continue, step through). The tab bar shows several files: main.m, maptowave.m (active), unipolar.m, ami.m, bipolarnrz.m, and bipolarrrz.m. The active file, maptowave.m, contains the following code:

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

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



The screenshot shows a MATLAB editor window with the same menu bar and toolbar as the previous image. The tab bar shows: main.m, maptowave.m, unipolar.m (active), ami.m, bipolarnrz.m, and bipolarrrz.m. 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);
```

```
main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m x
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);
```

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

```
main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m
1 % coding/bipolarrz.m
2 % Кодирование RZ:
3 function wave=bipolarrz(data)
4     data(data==0)=-1;
5     data=upsample(data,2);
6     wave=maptowave(data);
```

```
main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m x bipolarrrz.m x
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);

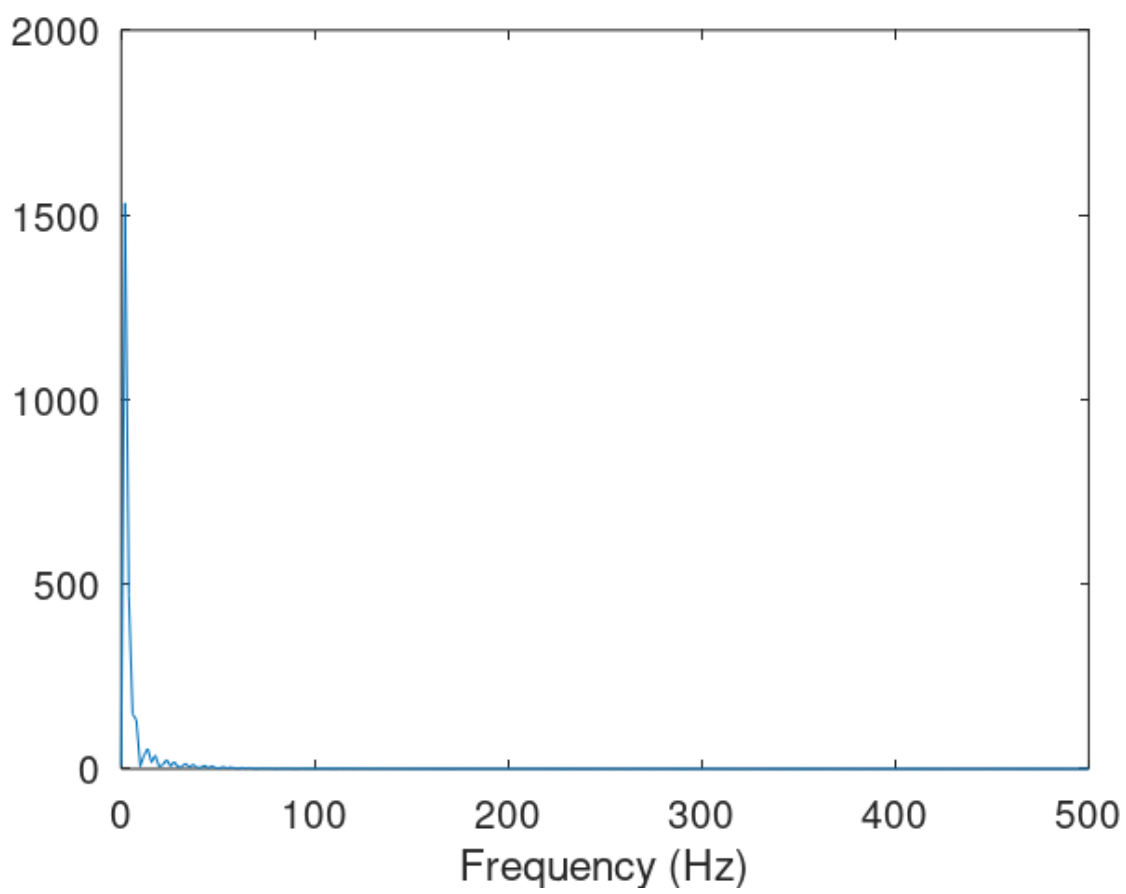
main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m x
1 % coding/diffmanc.m
2 % Дифференциальное манчестерское кодирование
3 function wave=diffmanc(data)
4     data=filter(1,[1 1],data);
5     data=mod(data,2);
6     wave=manchester(data);
```

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

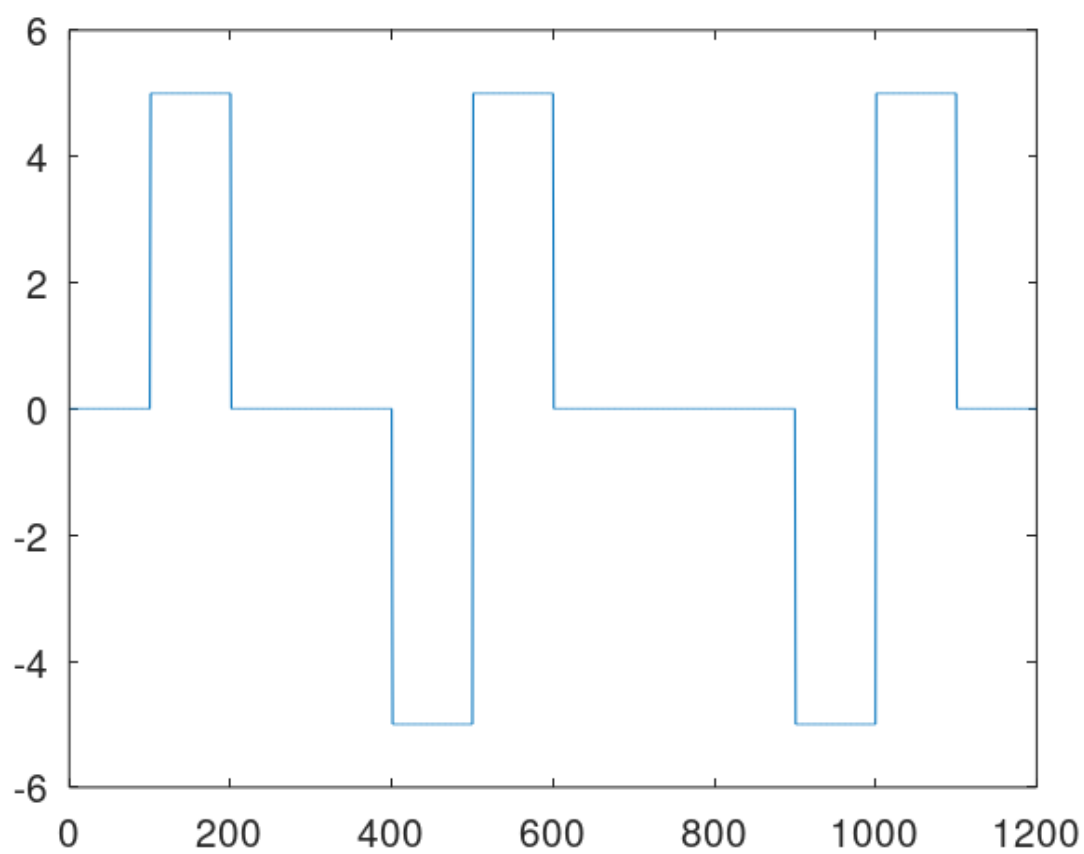
```
main.m x maptowave.m x unipolar.m x ami.m x bipolarnrz.m x bipolarrrz.m x
1 % calcspectre.m
2 % функция построения спектра сигнала:
3 function spectre = calcspectre(wave)
4     % Частота дискретизации (Гц):
5     Fd = 512;
6     Fd2 = Fd/2;
7     Fd3 = Fd/2 + 1;
8     X = fft(wave,Fd);
9     spectre = X.*conj(X)/Fd;
10    f = 1000*(0:Fd2)/Fd;
11    plot(f,spectre(1:Fd3));
12    xlabel('Frequency (Hz)');
13
```

7. Запустите главный скрипт `main.m`. В каталоге `signal` должны быть получены файлы с графиками кодированного сигнала, в каталоге `sync` файлы с графиками иллюстрирующими свойства самосинхронизации, в каталоге `spectre` — файлы с графиками спектров сигналов.

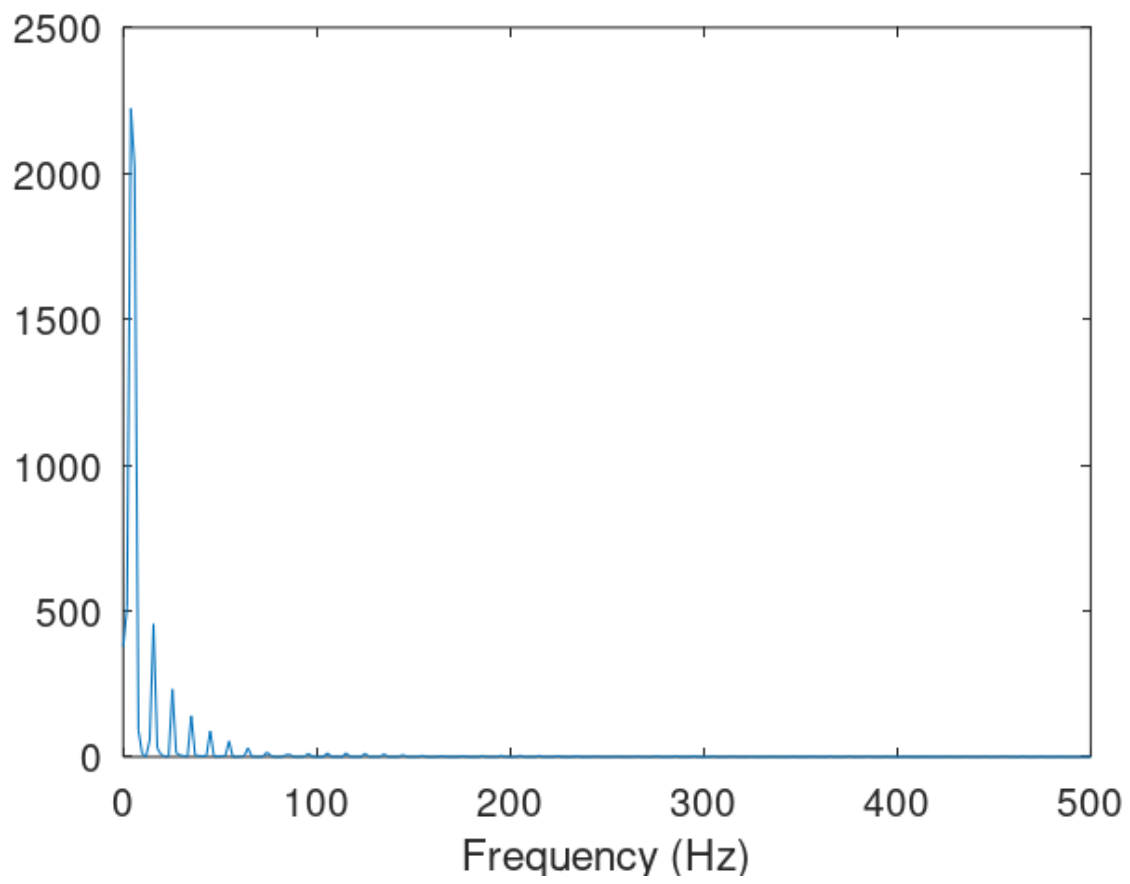
AMI



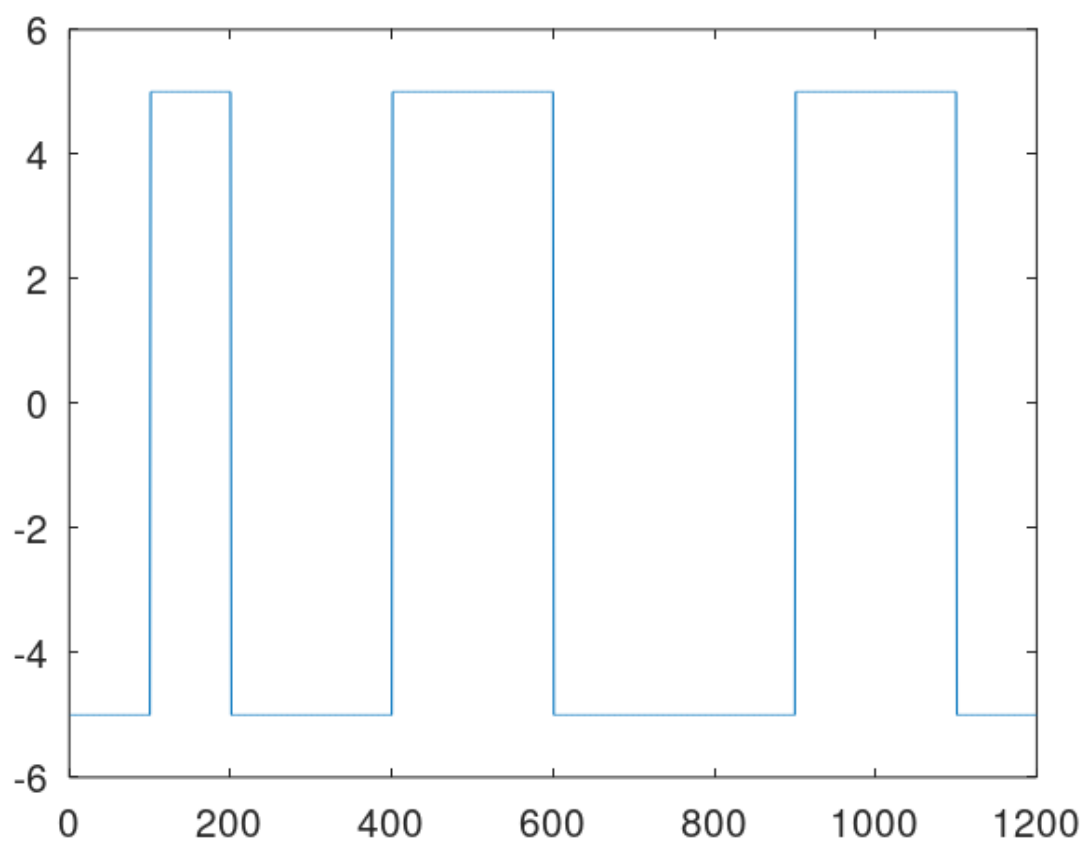
AMI



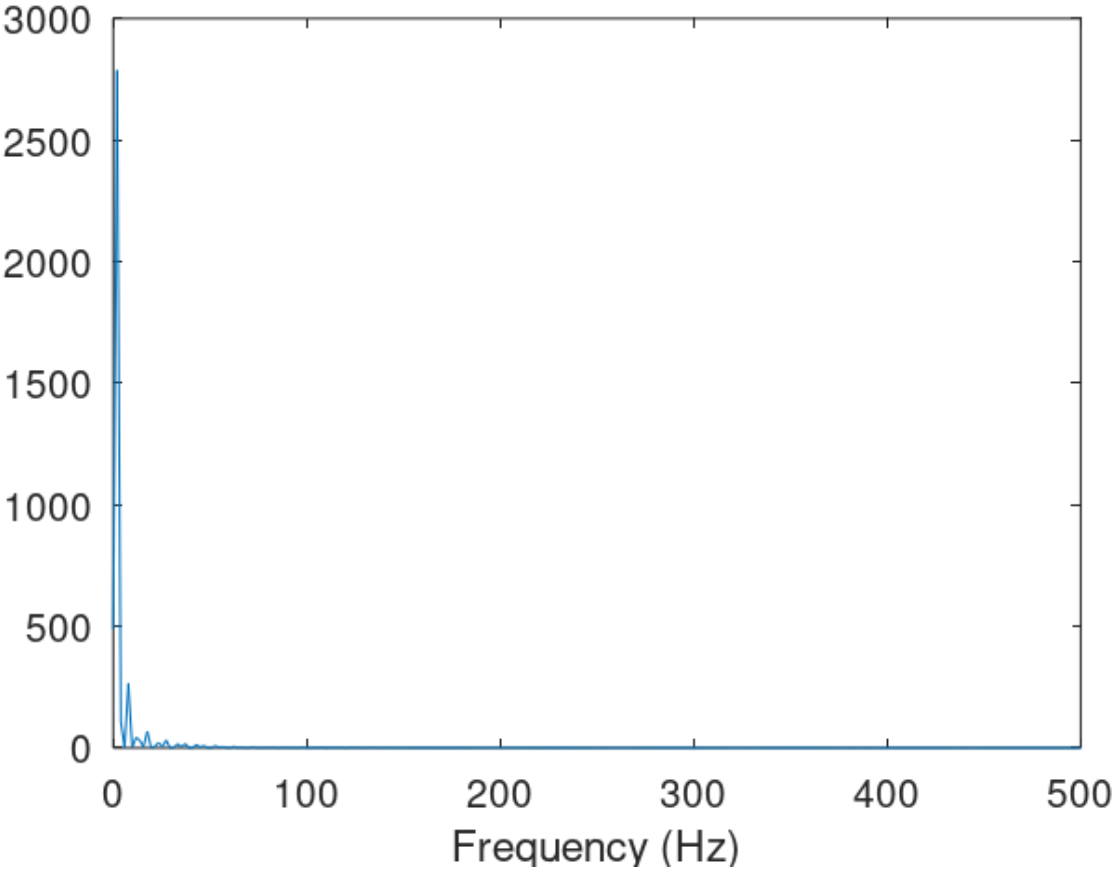
Bipolar Non-Return to Zero



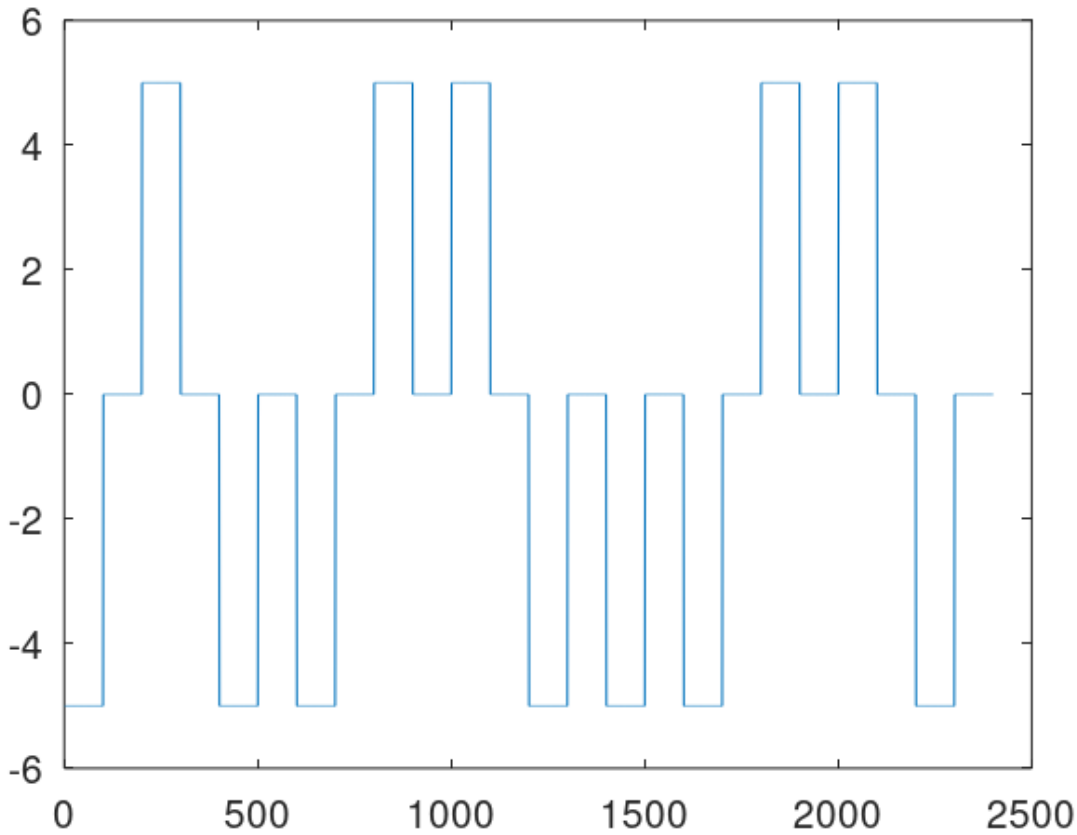
Bipolar Non-Return to Zero



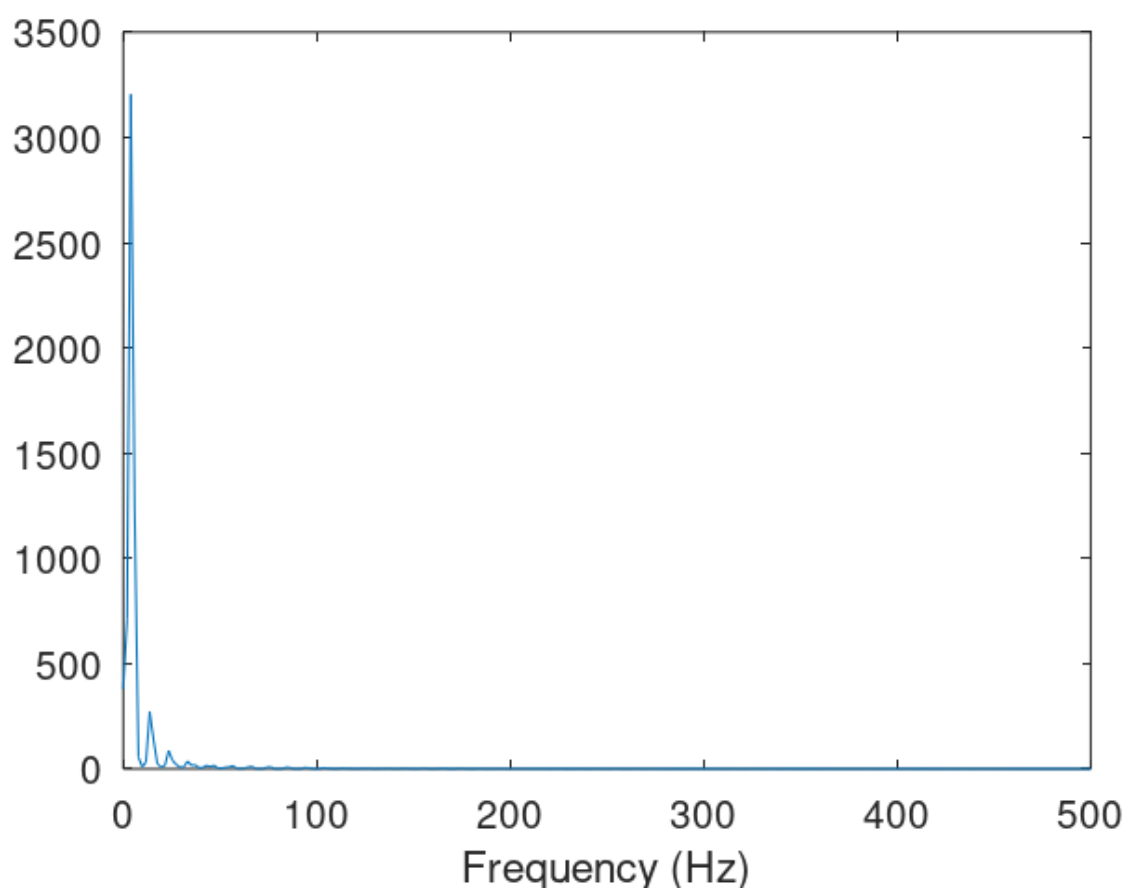
Bipolar Return to Zero



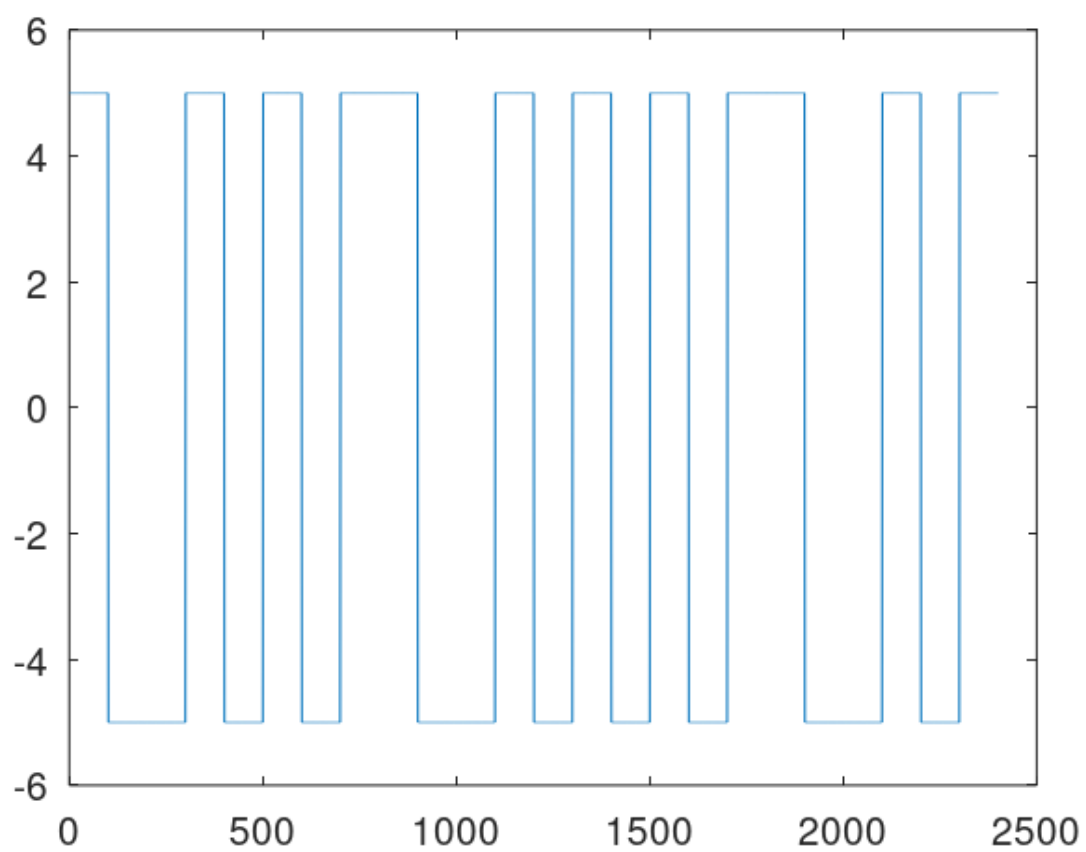
Bipolar Return to Zero



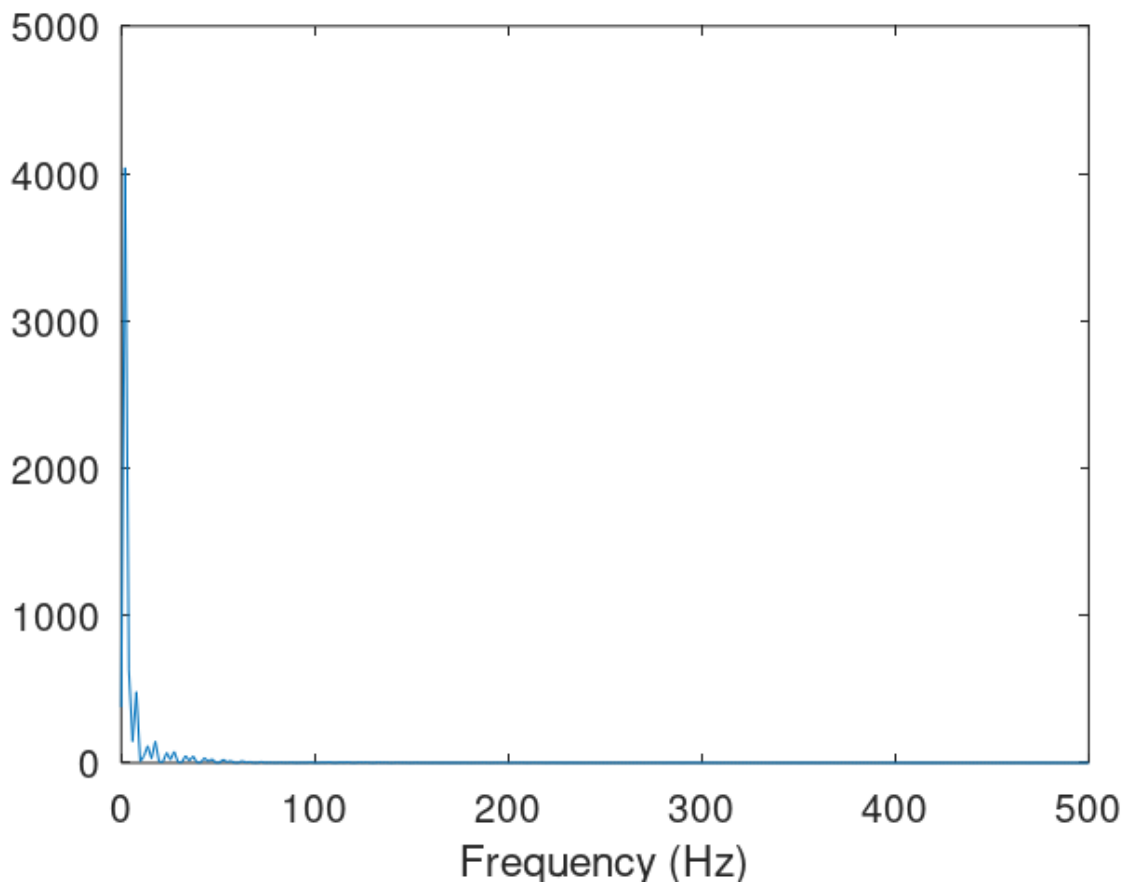
Differential Manchester



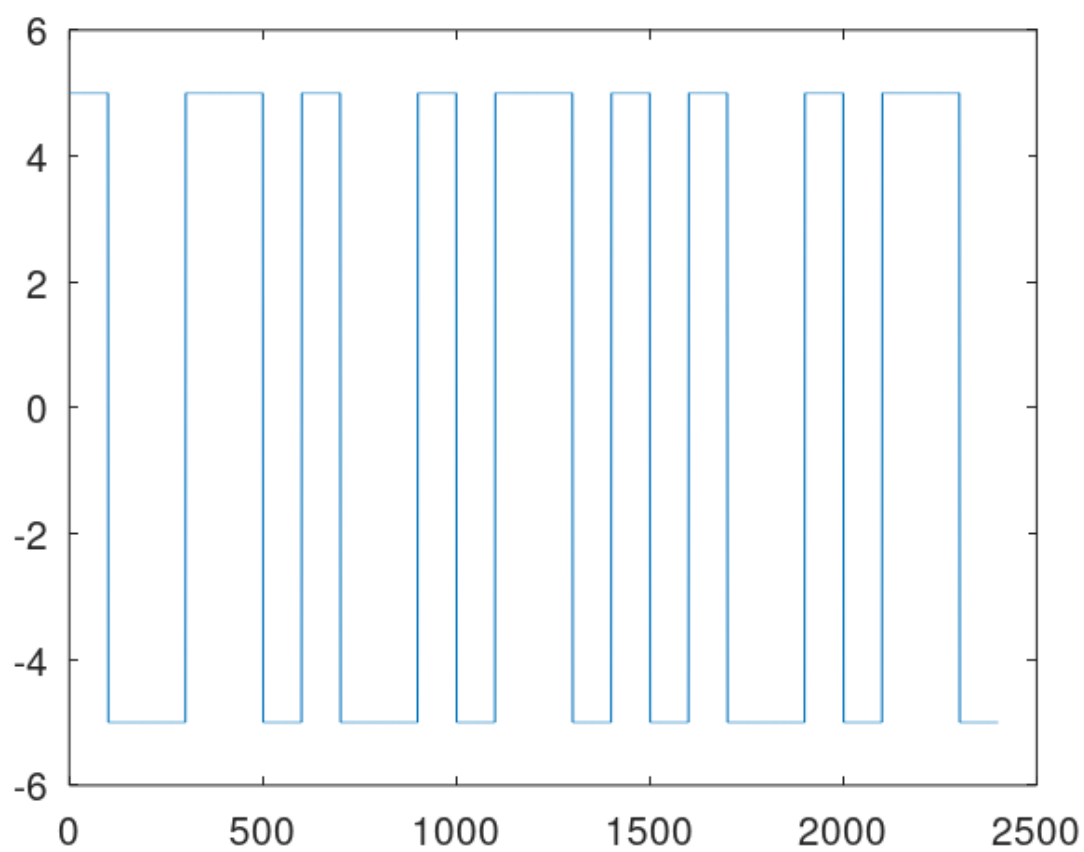
Differential Manchester



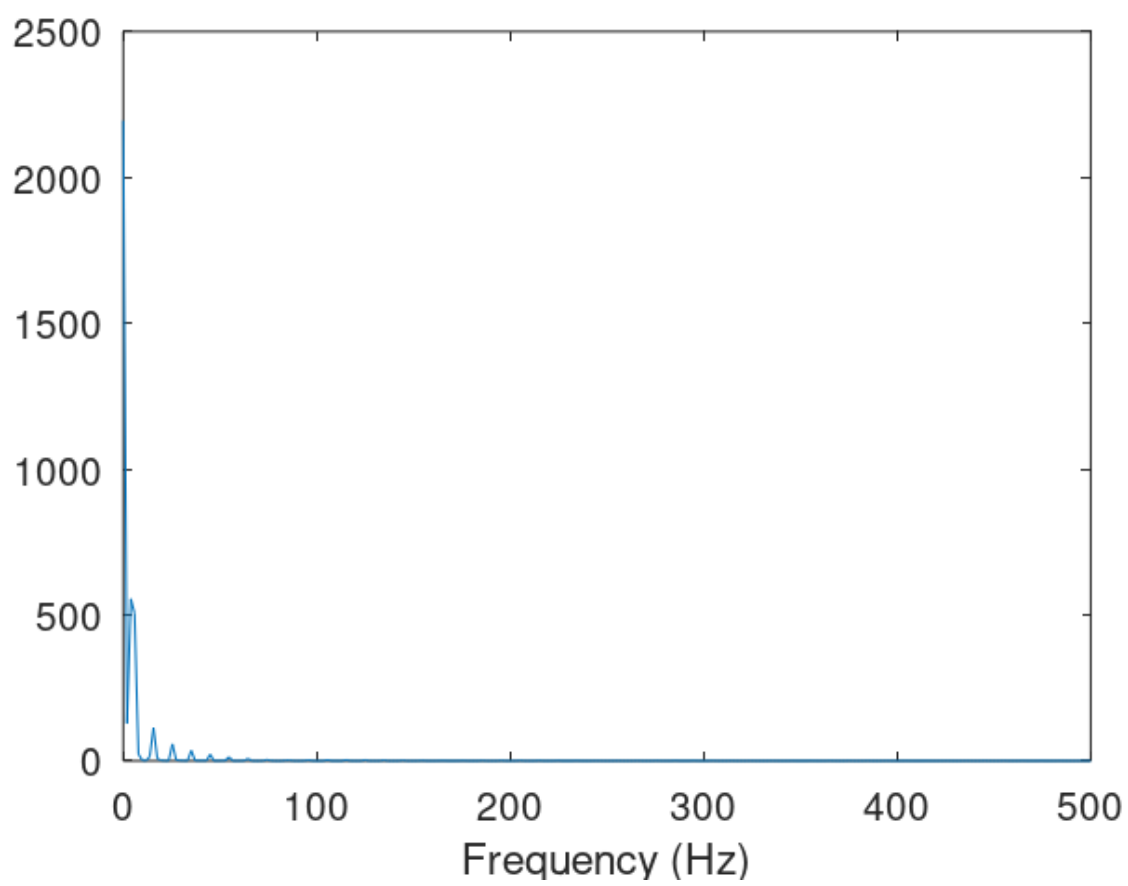
Manchester

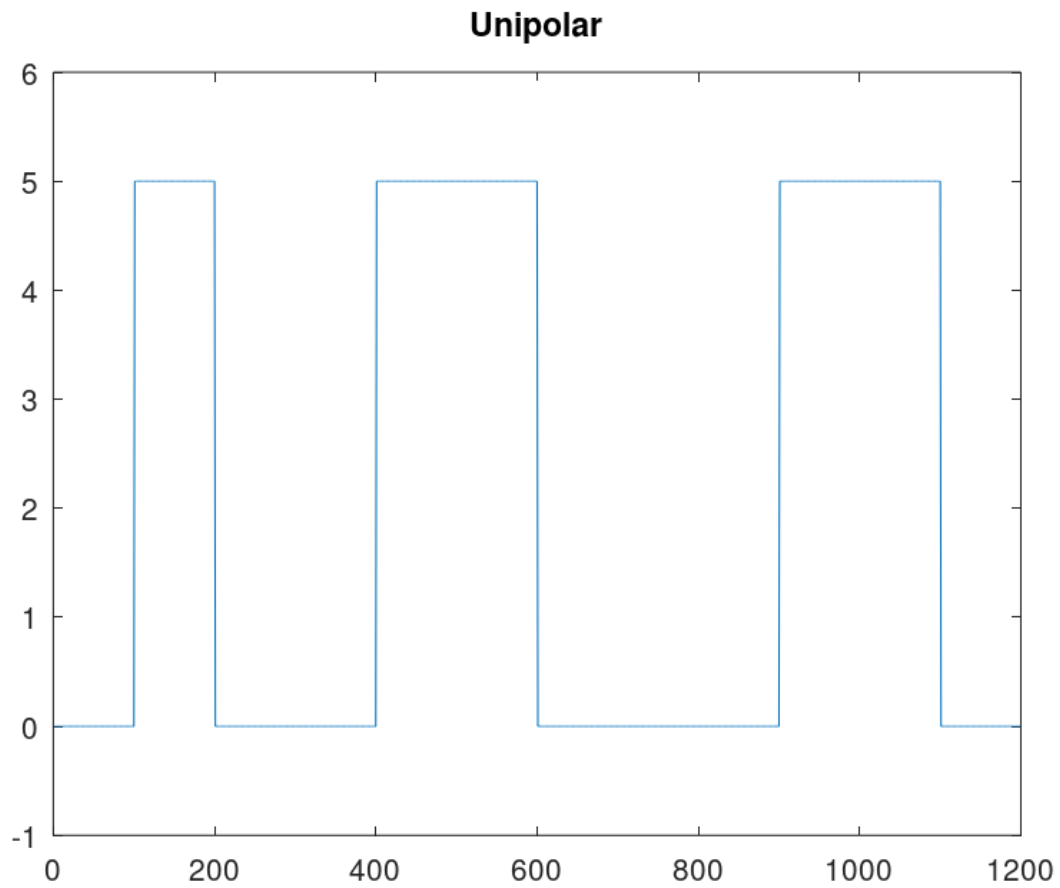


Manchester



Unipolar





Вывод

В ходе приобрел методов кодирования и модуляции сигналов с использованием языка программирования Octave. Демонстрация принципов модуляции сигнала на примере аналоговой амплитудной модуляции.