Лабораторная работа №1

Дисциплина: Сетевые технологии

Ибрахим Мохсейн Алькамаль

2025-12-20

Содержание I

Раздел 1

1. Информация

1.1 Докладчик

- Алькамаль Ибрахим Мохсейн Мохаммед Али
- Студент 3 курса
- Факультет: Фундаментальная информатика и информационные технологии
- Российский университет дружбы народов
- Email: 1032225432@pfur.ru
- GitHub: https://github.com/Ebrahimalkamal2027/study_2025-2026_nettech

1.2 Цели и задачи

- Изучение методов кодирования и модуляции сигналов в Octave
- Определение спектра и параметров сигнала
- Демонстрация принципов амплитудной модуляции
- Исследование свойства самосинхронизации сигнала

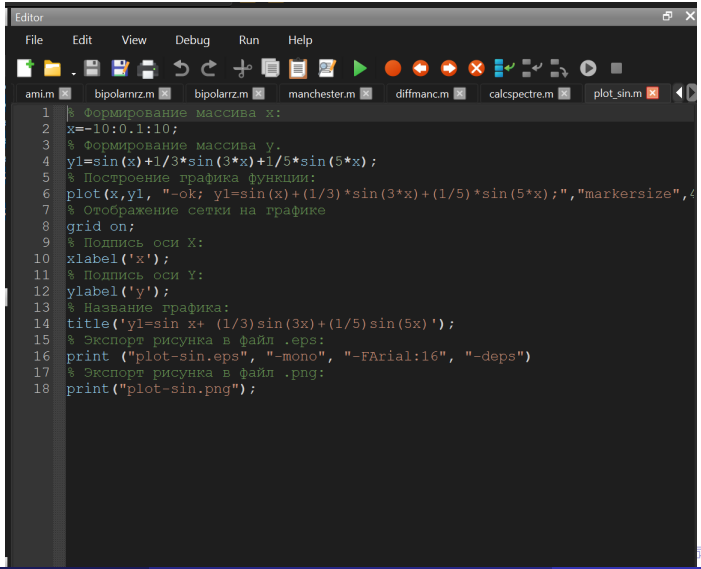
Раздел 2

2. Выполнение лабораторной работы

Раздел 3

3. Построение графиков в Octave

3.1 График y1


Editor
File
Edit
View
Debug
Run
Help
ami.m
bipolarnrz.m
bipolarrz.m
manchester.m
diffmanc.m
calcspectre.m
plot_sin.m
% Формирование массива x:
x=-10:0.1:10;
% Формирование массива y.
y1=sin(x)+1/3*sin(3*x)+1/5*sin(5*x);
% Построение графика функции:
plot(x,y1, "-ok; y1=sin(x)+(1/3)*sin(3*x)+(1/5)*sin(5*x);","markersize",
% Отображение сетки на графике
grid on;
% Подпись оси X:
xlabel('x');
% Подпись оси Y:
ylabel('y');
% Название графика:
title('y1=sin x+ (1/3)sin(3x)+(1/5)sin(5x)');
% Экспорт рисунка в файл .eps:
print ("plot-sin.eps", "-mono", "-FArial:16", "-deps")
% Экспорт рисунка в файл .png:
print("plot-sin.png");

3.2 График y1

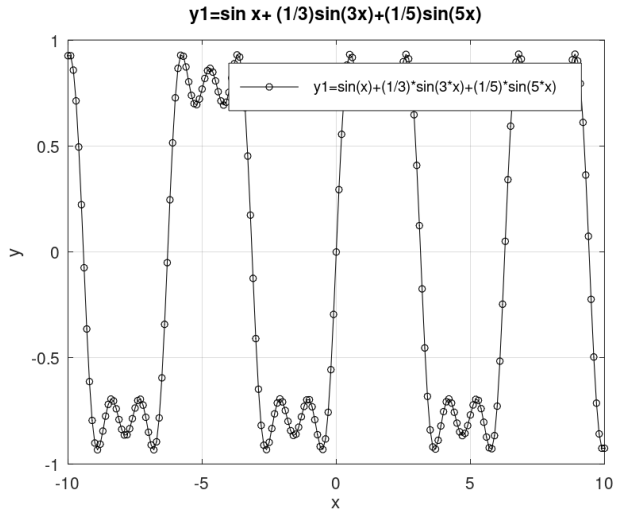


Рисунок 2: График функций y1 на интервале −10; 10

3.3 Проверка файлов

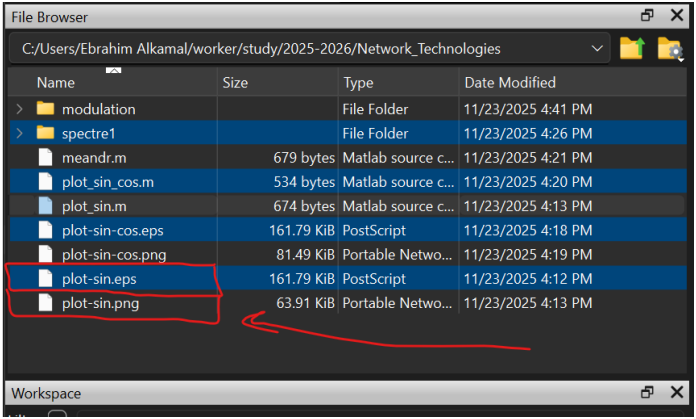
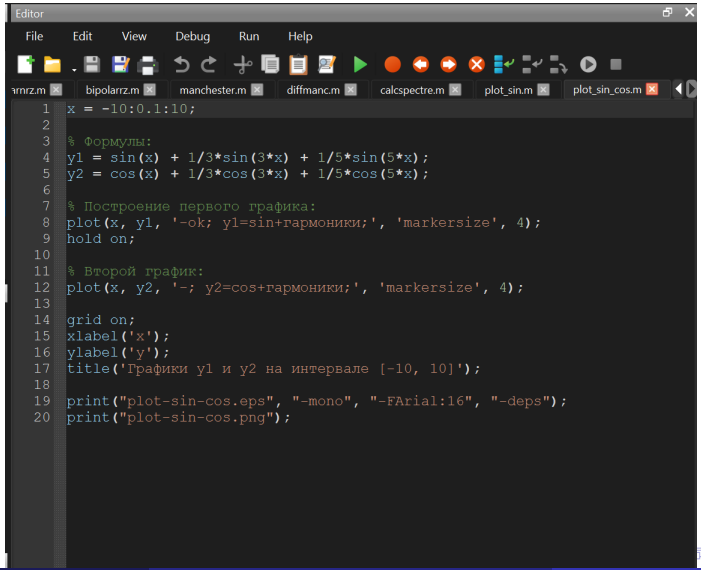


Рисунок 3: Файлы .eps, .png

3.4 Графики y1 и y2



3.5 Графики y1 и y2

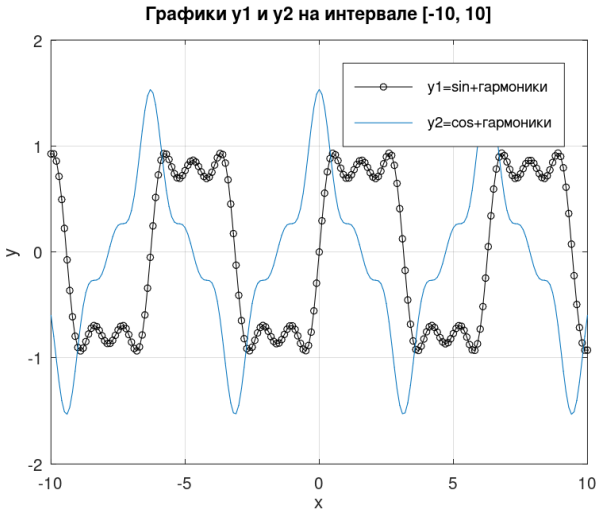


Рисунок 5: График функций y1 и y2 на интервале −10; 10

Раздел 4

4. Разложение импульсного сигнала в частичный ряд Фурье

4.1 Меандр через косинус

- Создадим новый сценарий meandr.m. В коде зададим начальные значения. Вычислим амплитуду гармоник и заполним массивы гармоник и элементов ряда. Далее задаём массив значений гармоник массив элементов ряда. Также экспортируем полученный график в файл в формате .png

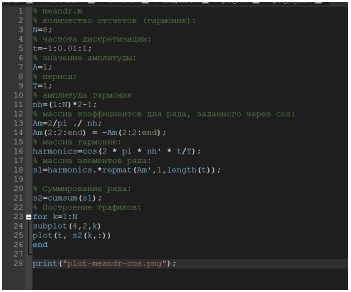


Рисунок 6: Листинг файла meandr.m

4.2 Результат

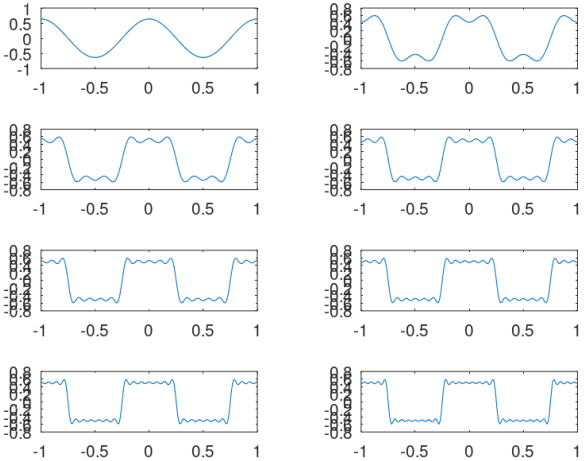


Рисунок 7: Меандр через косинусы

4.3 Меандр через синус

```
% 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));

% Сумирование ряда:
s2=cumsum(s1);
% Построение графиков:
for k=1:N
  subplot(4,2,k)
  plot(t, s2(k,:))
end

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

Рисунок 8: Листинг файла meandr.m

4.4 Результат

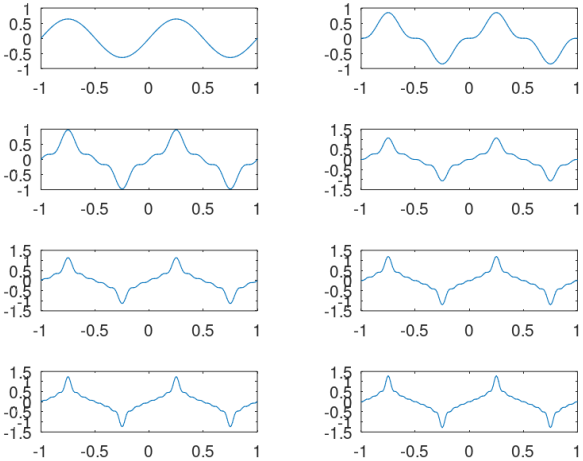


Рисунок 9: Меандр через синусы

Раздел 5

5. Определение спектра и параметров сигнала

5.1 Сигналы разной 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';
```

Рисунок 10: Листинг файла spectre.m

5.2 Результат

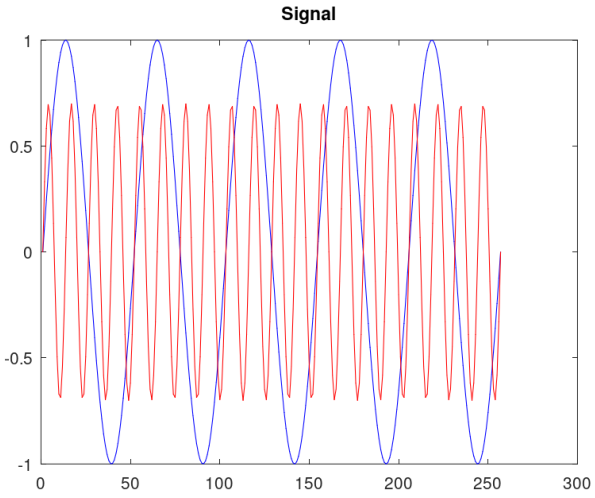


Рисунок 11: Графики сигналов разной частоты

5.3 Спектры сигналов

```
% 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';
```

Рисунок 12: Листинг файла spectre.m

5.4 График спектра

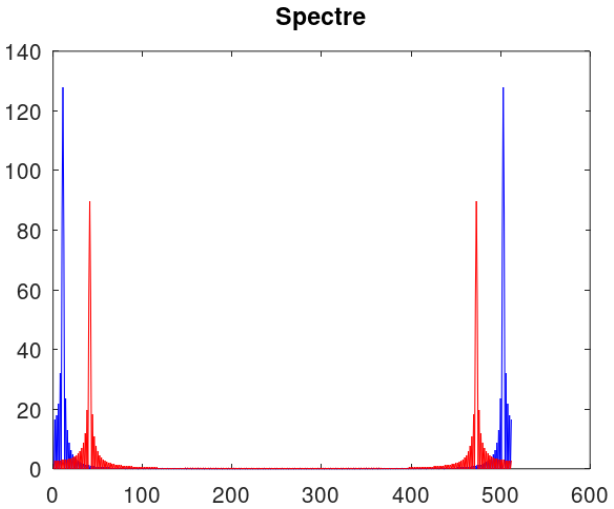


Рисунок 13: График спектра синусоидальных сигналов

5.5 Скорректированный график

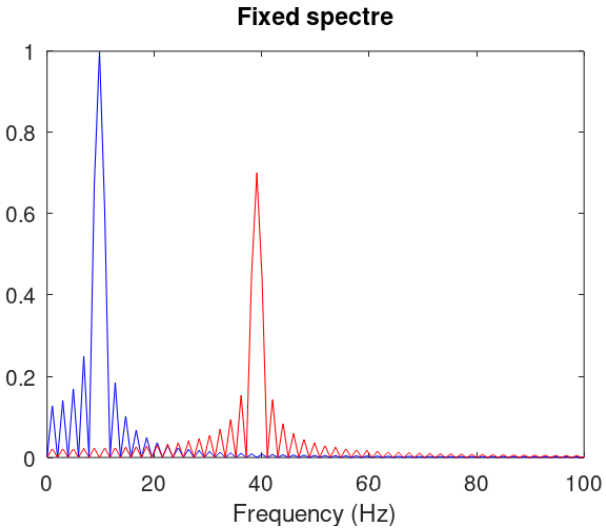


Рисунок 14: Исправленный график спектров синусоидальных сигналов

5.6 Спектр суммы

spectre_sum.m

```
% spectr_sum/spectre_sum.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;
% 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 спектраX
% 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';
```

Рисунок 15: Листинг файла spectre_sum.m

5.7 Сумма

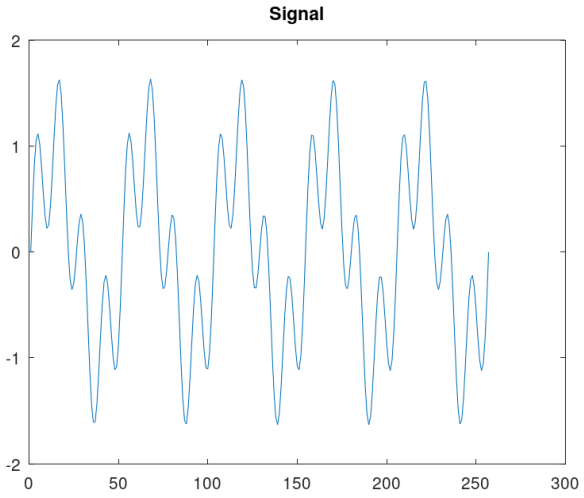


Рисунок 16: Суммарный сигнал

5.8 Спектр суммы

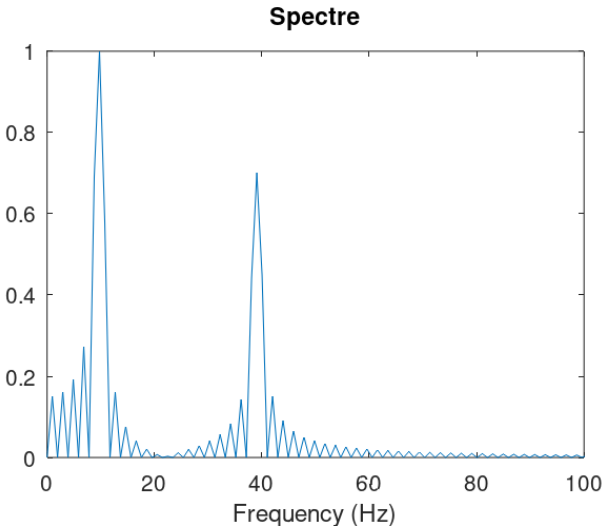


Рисунок 17: Спектр суммарного сигнала

Раздел 6

6. Амплитудная модуляция

6.1 Амплитудная модуляция

am.m

```
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]);

title('Spectre');
xlabel('Frequency (Hz)');
print 'spectre/am.png';
```

Рисунок 18: Листинг файла am.m

6.2 Результат

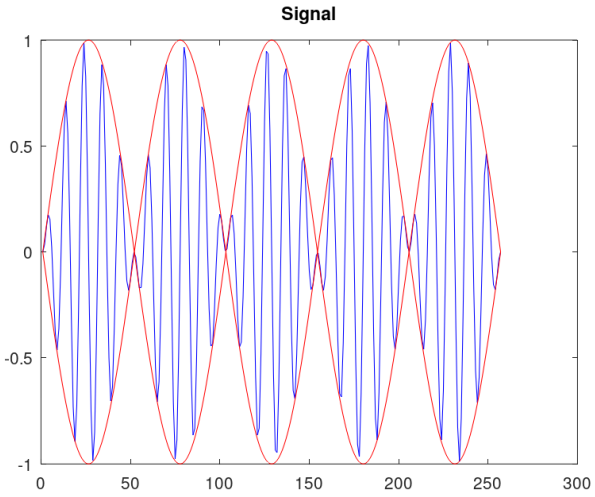


Рисунок 19: Сигнал и огибающая при амплитудной модуляции

6.3 Результат

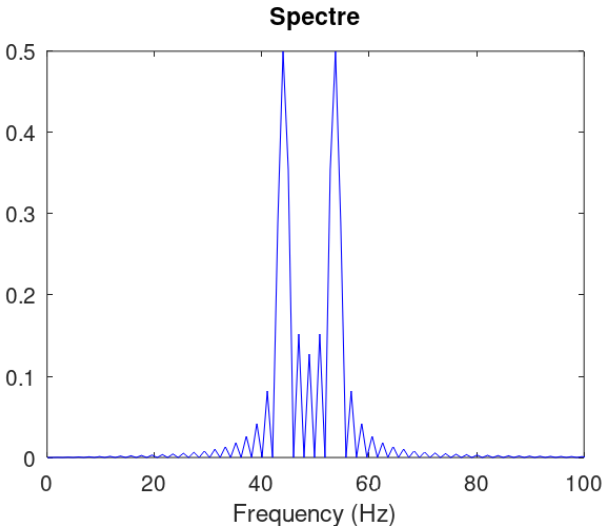


Рисунок 20: Спектр сигнала при амплитудной модуляции

Раздел 7

7. Кодирование сигнала. Исследование свойства самосинхронизации сигнала

7.1 Подготовка

- В рабочем каталоге создадим 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 проверяем, установлен ли пакет расширений signal: pkg list. Так как он не установлен, то устанавливаем его: pkg list -forge и pkg install control signal

```
>> pkg list
Package Name          | Version | Installation directory
----------------------+---------+-----------------------
               audio  |   2.0.9 | ...\Octave-10.3.0\mingw64\share\octave\packages\audio-2.0.9
             biosig   |   3.9.0 | ...\Octave-10.3.0\mingw64\share\octave\packages\biosig-3.9.0
             cfitsio  |   0.0.7 | ...\Octave-10.3.0\mingw64\share\octave\packages\cfitsio-0.0.7
               coder  |  1.10.1 | ...\Octave-10.3.0\mingw64\share\octave\packages\coder-1.10.1
      communications  |   1.2.7 | ...\mingw64\share\octave\packages\communications-1.2.7
             control  |   4.1.3 | ...\Octave-10.3.0\mingw64\share\octave\packages\control-4.1.3
    data-smoothing    |   1.3.0 | ...\mingw64\share\octave\packages\data-smoothing-1.3.0
            database  |   2.4.4 | ...\Octave-10.3.0\mingw64\share\octave\packages\database-2.4.4
           dataframe  |   1.2.0 | ...\mingw64\share\octave\packages\dataframe-1.2.0
               dicom  |   0.6.1 | ...\Octave-10.3.0\mingw64\share\octave\packages\dicom-0.6.1
           financial  |   0.5.4 | ...\mingw64\share\octave\packages\financial-0.5.4
             fl-core  |   1.0.2 | ...\Octave-10.3.0\mingw64\share\octave\packages\fl-core-1.0.2
fuzzy-logic-toolkit   |   0.6.2 | ...\mingw64\share\octave\packages\fuzzy-logic-toolkit-0.6.2
                  ga  |  0.10.4 | ...\Octave-10.3.0\mingw64\share\octave\packages\ga-0.10.4
             general  |   2.1.3 | ...\Octave-10.3.0\mingw64\share\octave\packages\general-2.1.3
      generate_html   |   0.3.3 | ...\mingw64\share\octave\packages\generate_html-0.3.3
            geometry  |   4.1.0 | ...\Octave-10.3.0\mingw64\share\octave\packages\geometry-4.1.0
                 gsl  |   2.1.1 | ...\Octave-10.3.0\mingw64\share\octave\packages\gsl-2.1.1
               image  |  2.18.1 | ...\Octave-10.3.0\mingw64\share\octave\packages\image-2.18.1
  image-acquisition   |   0.3.3 | ...\mingw64\share\octave\packages\image-acquisition-0.3.3
instrument-control    |   0.9.5 | ...\mingw64\share\octave\packages\instrument-control-0.9.5
            interval  |   3.2.1 | ...\Octave-10.3.0\mingw64\share\octave\packages\interval-3.2.1
                  io  |   2.7.0 | ...\Octave-10.3.0\mingw64\share\octave\packages\io-2.7.0
     linear-algebra   |   2.2.3 | ...\mingw64\share\octave\packages\linear-algebra-2.2.3
                lssa  |   0.1.4 | ...\Octave-10.3.0\mingw64\share\octave\packages\lssa-0.1.4
              ltfat   |   2.6.0 | ...\Octave-10.3.0\mingw64\share\octave\packages\ltfat-2.6.0
             mapping  |   1.4.3 | ...\Octave-10.3.0\mingw64\share\octave\packages\mapping-1.4.3
             matgeom  |   1.2.4 | ...\Octave-10.3.0\mingw64\share\octave\packages\matgeom-1.2.4
      miscellaneous   |   1.3.1 | ...\mingw64\share\octave\packages\miscellaneous-1.3.1
```

7.2 Файл main.m

main.m

```
% coding/main.m
% Подключение пакета signal:
pkg load signal;
% Входная кодовая последовательность:
data=[0 1 0 0 1 1 0 0 0 1 1 0];
% Входная кодовая последовательность для проверки свойства самосинхрон
data_sync=[0 0 0 0 0 0 0 1 1 1 1 1 1 1];
% Входная кодовая последовательность для построения спектра сигнала:
data_spectre=[0 1 0 1 0 1 0 1 0 1 0 1 0 1];
% Создание каталогов signal, sync и spectre для размещения графиков:
mkdir 'signal';
mkdir 'sync';
mkdir 'spectre';
axis("auto");
```

Рисунок 22: Задаем входные кодовые последовательности

7.3 Файл main.m

```
% 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';
```

Рисунок 23: Вызовы функций для посторения модуляций кодированных сигналов кодовой последовательности data

7.4 Файл main.m

main.m

```
47 % Униполярное кодирование
48 wave=unipolar(data_sync);
49 plot(wave);
50 ylim([-1 6]);
51 title('Unipolar');
52 print 'sync/unipolar.png';
53 % Кодирование AMI
54 wave=ami(data_sync);
55 plot(wave)
56 title('AMI');
57 print 'sync/ami.png';
58 % Кодирование NRZ
59 wave=bipolarnrz(data_sync);
60 plot(wave);
61 title('Bipolar Non-Return to Zero');
62 print 'sync/bipolarnrz.png';
63 % Кодирование RZ
64 wave=bipolarrz(data_sync);
65 plot(wave)
66 title('Bipolar Return to Zero');
67 print 'sync/bipolarrz.png';
68 % Манчестерское кодирование
69 wave=manchester(data_sync);
70 plot(wave)
71 title('Manchester');
72 print 'sync/manchester.png';
73 % Дифференциальное манчестерское кодирование
74 wave=diffmanc(data_sync);
75 plot(wave)
76 title('Differential Manchester');
77 print 'sync/diffmanc.png';
```

Рисунок 24: Вызовы функций для посторения модуляций кодированных сигналов кодовой последовательности data_sync

7.5 Файл main.m

```
% 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';

```

Рисунок 25: Вызовы функций для посторения графиков спектров

7.6 Файл maptowave.m

main.m maptowave.m

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

Рисунок 26: Листинг файла maptowave.m

7.7 Файл unipolar.m

unipolar.m

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

Рисунок 27: Листинг файла unipolar.m

7.8 Файл ami.m

unipolar.m ami.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);
end
```

Рисунок 28: Листинг файла ami.m

7.9 Файл bipolarnrz.m

unipolar.m | ami.m | bipolarnrz.m

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

Рисунок 29: Листинг файла bipolarnrz.m

7.10 Файл bipolarrz.m

bipolarrz.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);
7  end
```

Рисунок 30: Листинг файла bipolarrz.m

7.11 Файл manchester.m

bipolarrz.m manchester.m

```
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);
8  end
```

Рисунок 31: Листинг файла manchester.m

7.12 Файл diffmanc.m

diffmanc.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);
```

Рисунок 32: Листинг файла diffmanc.m

7.13 Файл calcspectre.m

diffmanc.m | calcspectre.m

```
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 end
```

Рисунок 33: Листинг файла calcspectre.m

7.14 График кодированного сигнала

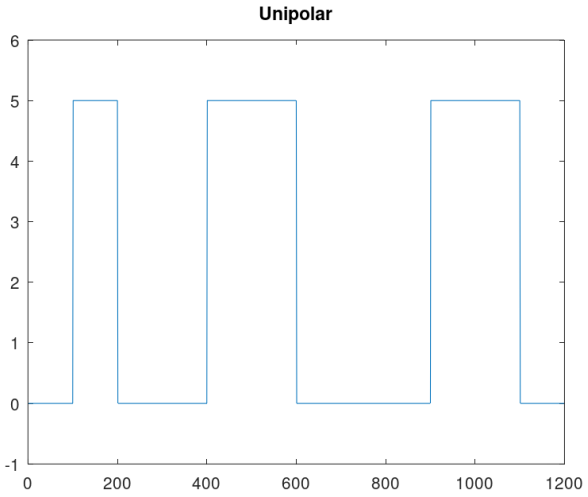


Рисунок 34: Униполярное кодирование

7.15 График кодированного сигнала

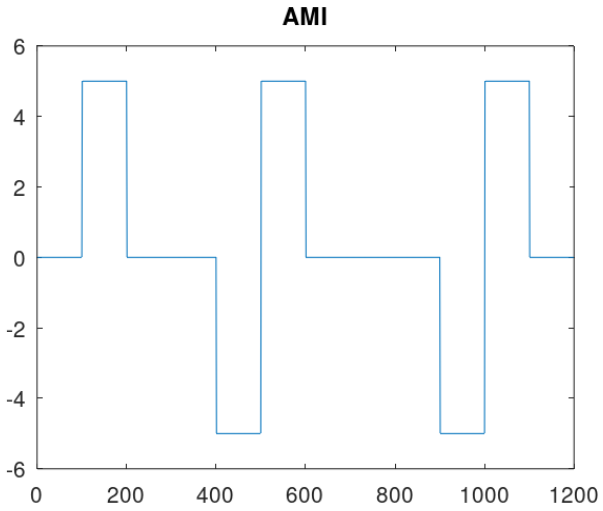


Рисунок 35: Кодирование AMI

7.16 График кодированного сигнала

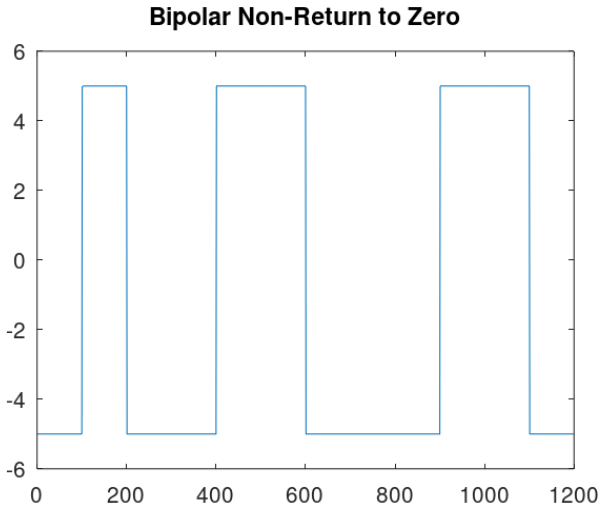


Рисунок 36: Кодирование NRZ

7.17 График кодированного сигнала

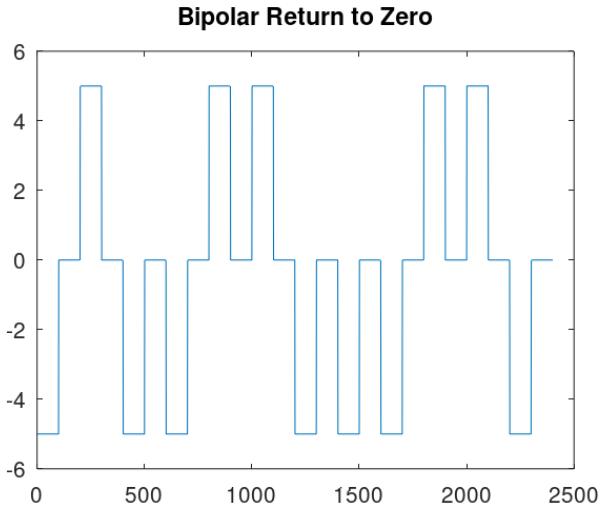


Рисунок 37: Кодирование RZ

7.18 График кодированного сигнала

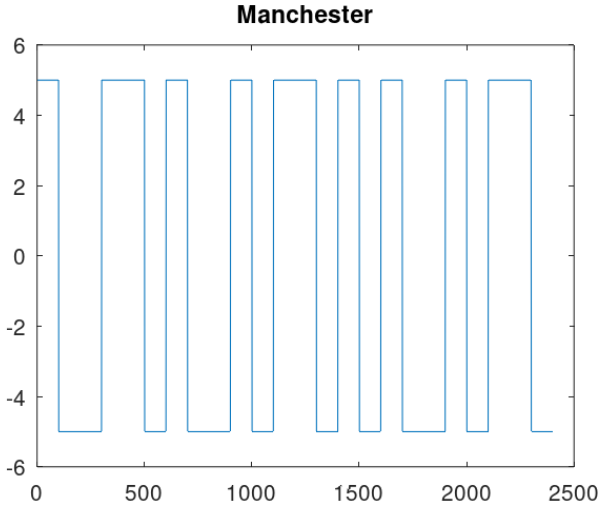


Рисунок 38: Манчестерское кодирование

7.19 График кодированного сигнала

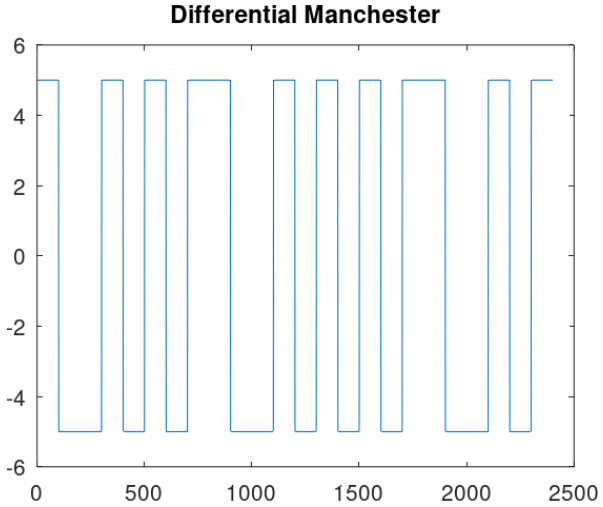


Рисунок 39: Дифференциальное манчестерское кодирование

7.20 Иллюстрация свойства самосинхронизации

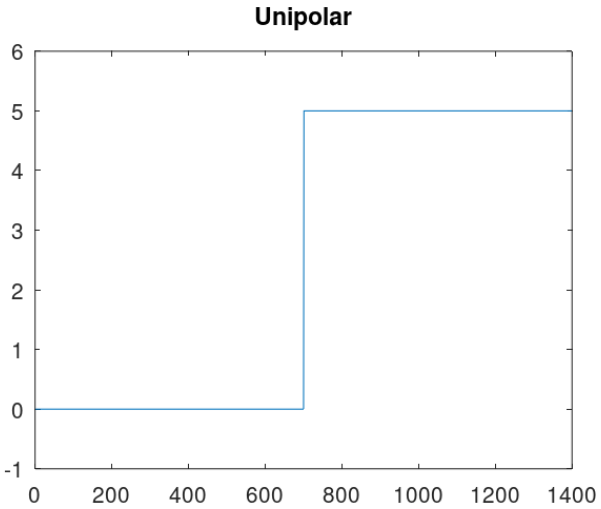


Рисунок 40: Униполярное кодирование: нет самосинхронизации

7.21 Иллюстрация свойства самосинхронизации

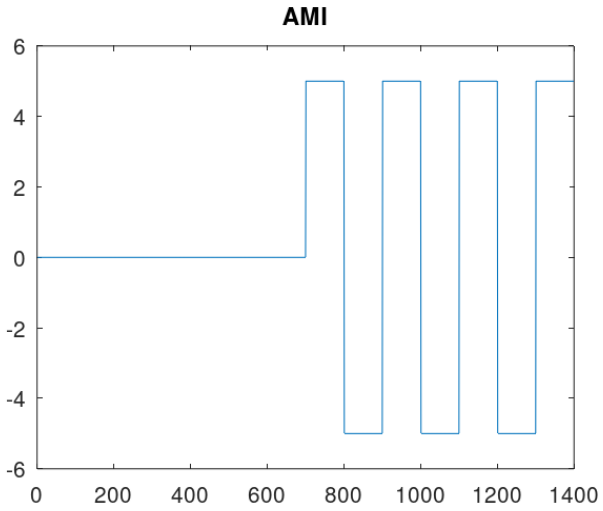


Рисунок 41: Кодирование AMI: самосинхронизация при наличии сигнала

7.22 Иллюстрация свойства самосинхронизации

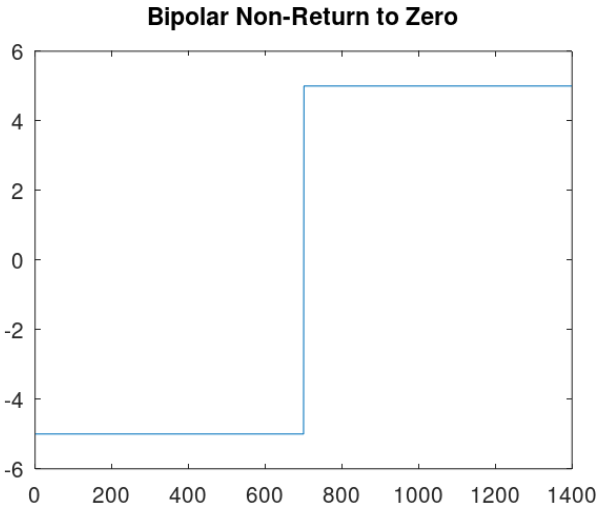


Рисунок 42: Кодирование NRZ: нет самосинхронизации

7.23 Иллюстрация свойства самосинхронизации

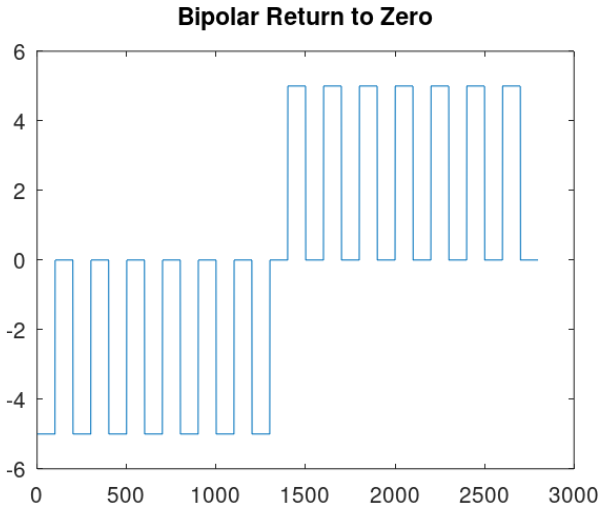


Рисунок 43: Кодирование RZ: есть самосинхронизация

7.24 Иллюстрация свойства самосинхронизации

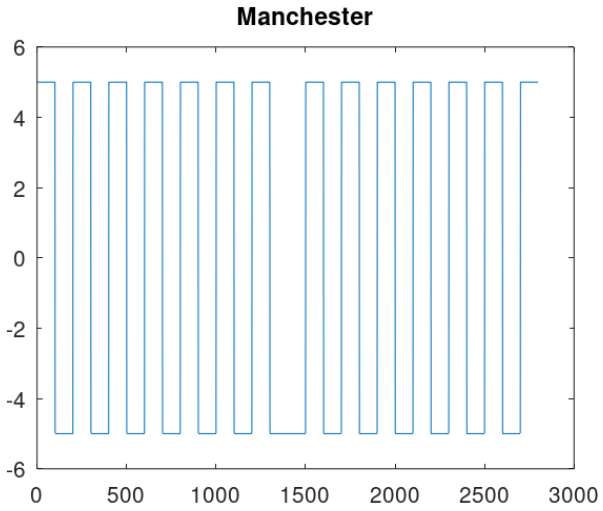


Рисунок 44: Манчестерское кодирование: есть самосинхронизация

7.25 Иллюстрация свойства самосинхронизации

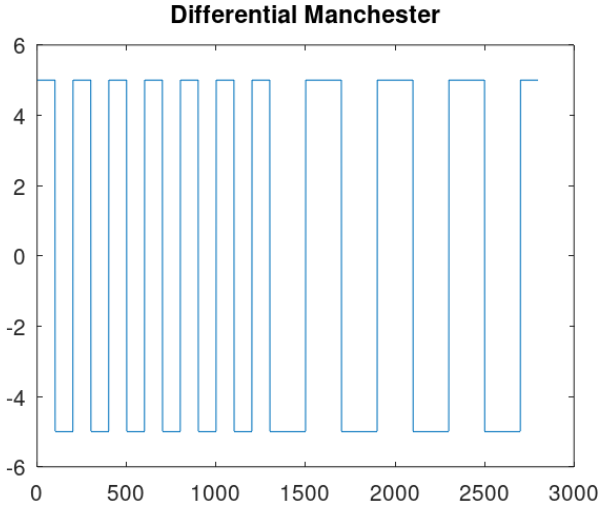


Рисунок 45: Дифференциальное манчестерское кодирование: есть самосинхронизация

7.26 Графики спектра сигнала

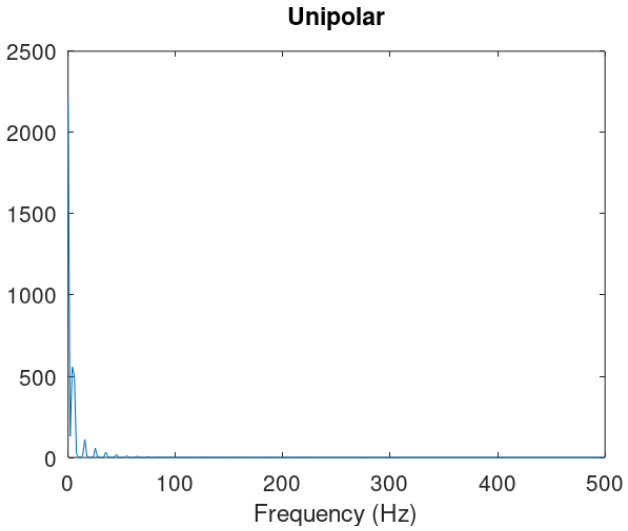


Рисунок 46: Униполярное кодирование: спектр сигнала

7.27 Графики спектра сигнала

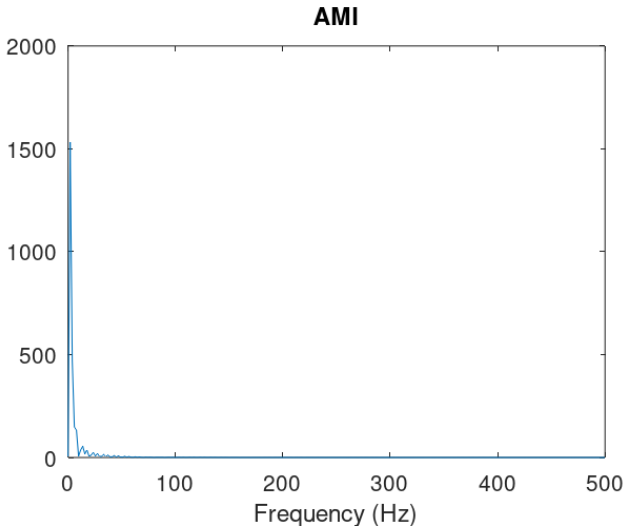


Рисунок 47: Кодирование AMI: спектр сигнала

7.28 Графики спектра сигнала

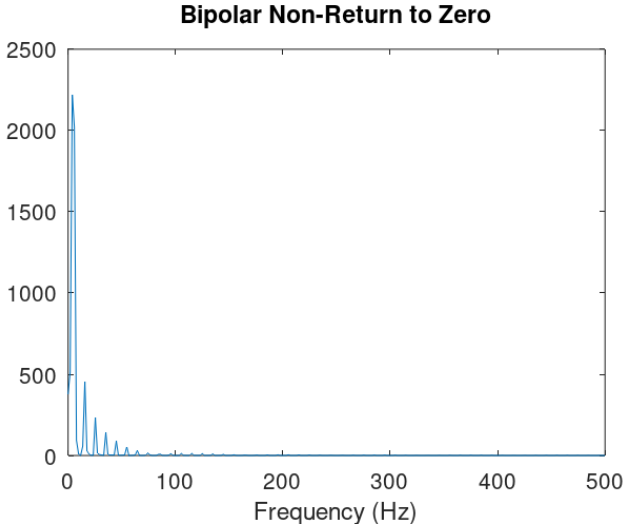


Рисунок 48: Кодирование NRZ: спектр сигнала

7.29 Графики спектра сигнала

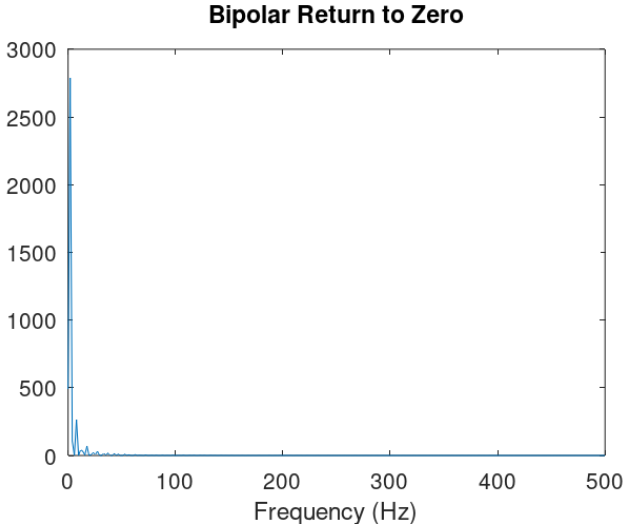


Рисунок 49: Кодирование RZ: спектр сигнала

7.30 Графики спектра сигнала

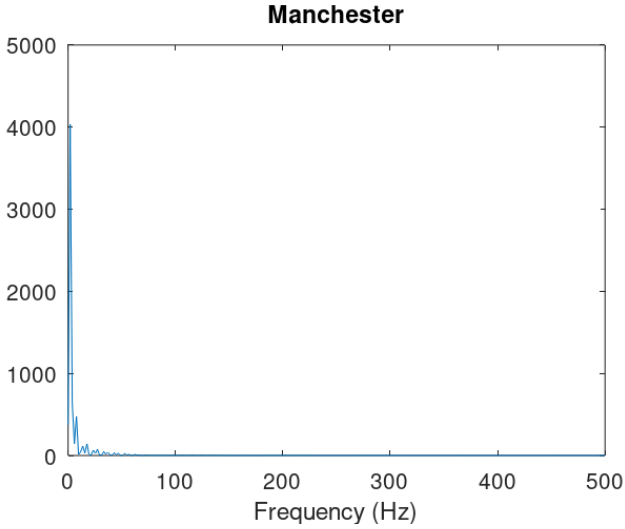


Рисунок 50: Манчестерское кодирование: спектр сигнала

7.31 Графики спектра сигнала

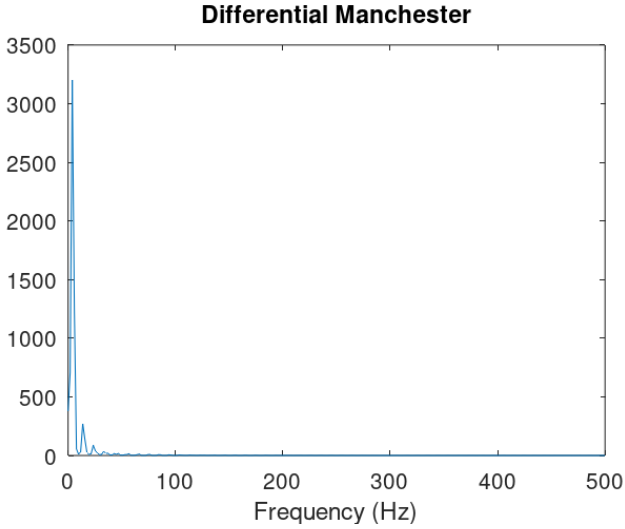


Рисунок 51: Дифференциальное манчестерское кодирование: спектр сигнала

7.32 Вывод

- В ходе выполнения данной лабораторной работы я изучила методы кодирования и модуляции сигналов с помощью высокоуровнего языка программирования Octave. Определила спектр и параметры сигнала. Продемонстрировала принципы модуляции сигнала на примере аналоговой амплитудной модуляции. Исследовала свойства самосинхронизации сигнала.