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: Юсупов Ш

Ст.номер:1032205329

Группа: НПИбд-02-20

МОСКВА

2022 г.

ЦЕЛЬ РАБОТЫ:

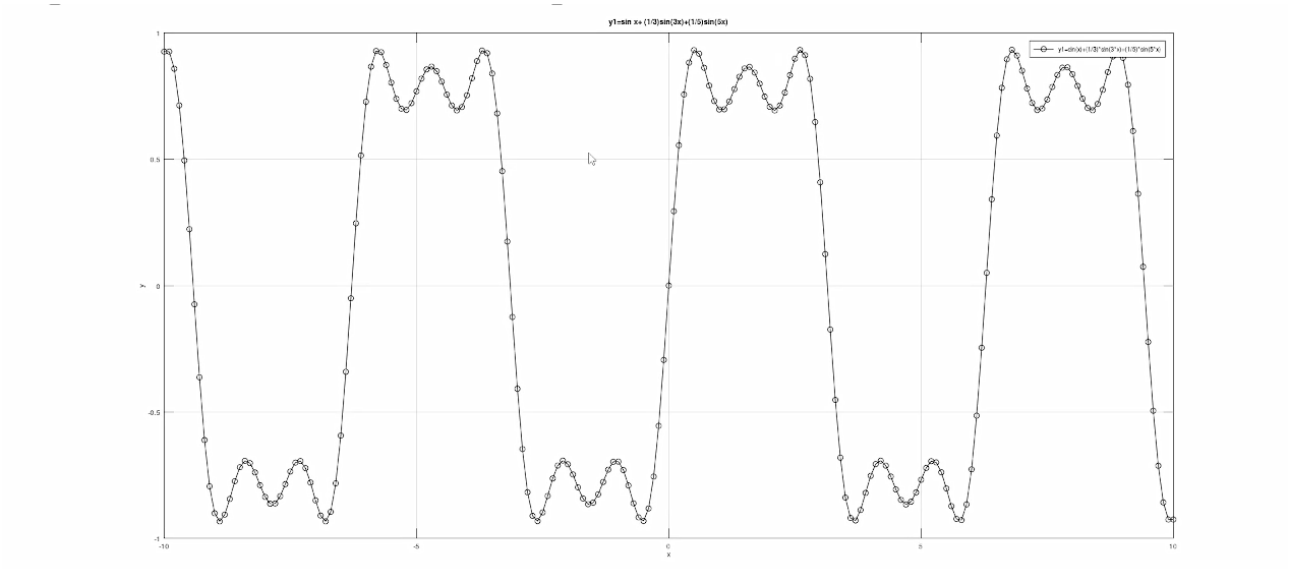
Изучение методов кодирования и модуляции сигналов с помощью высокоуровневого языка программирования Octave. Определение спектра и параметров сигнала. Демонстрация принципов модуляции сигнала на примере аналоговой амплитудной модуляции. Исследование свойства самосинхронизации сигнала

1. Запускаем Octave с оконным интерфейсом. Создаем новый файл plot_sin.m. и вводим туда листинг по построению графика функции $y = \sin x + 1\ 3 \sin 3x + 1\ 5 \sin 5x$ на интервале [−10; 10]:

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",4)
% 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", "-depsc")
% Экспорт рисунка в файл .png:
print("plot-sin.png");
```

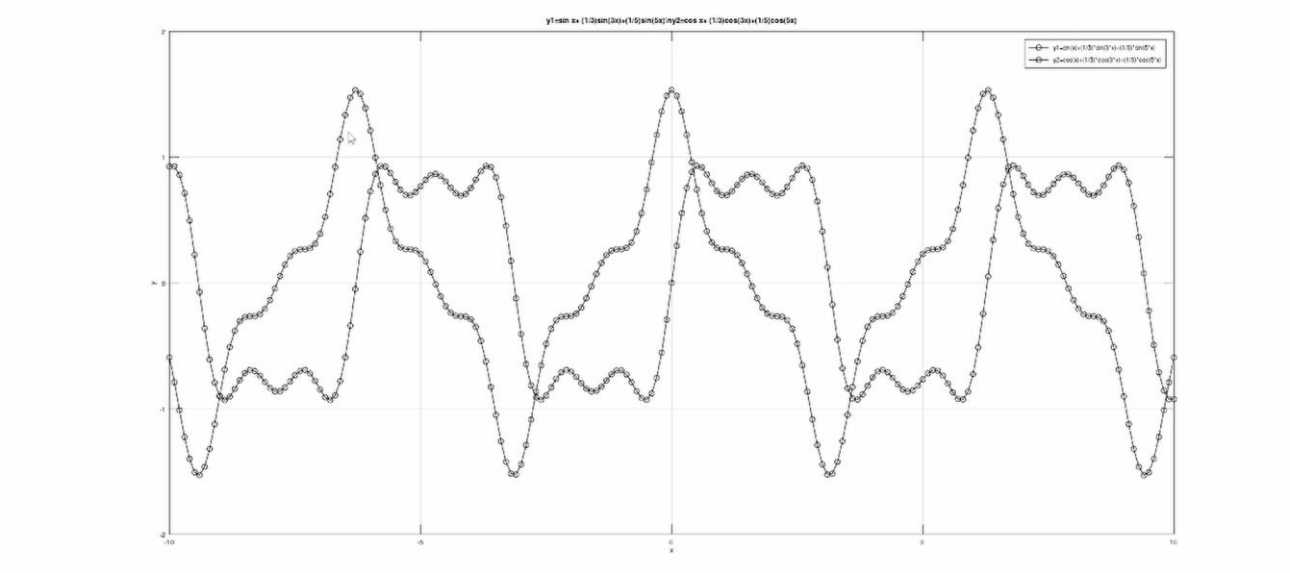
2. Запускаем его используя команду F5



3. Меняем код, так, чтобы на одной оси показывал два графика

plot_sincos.m

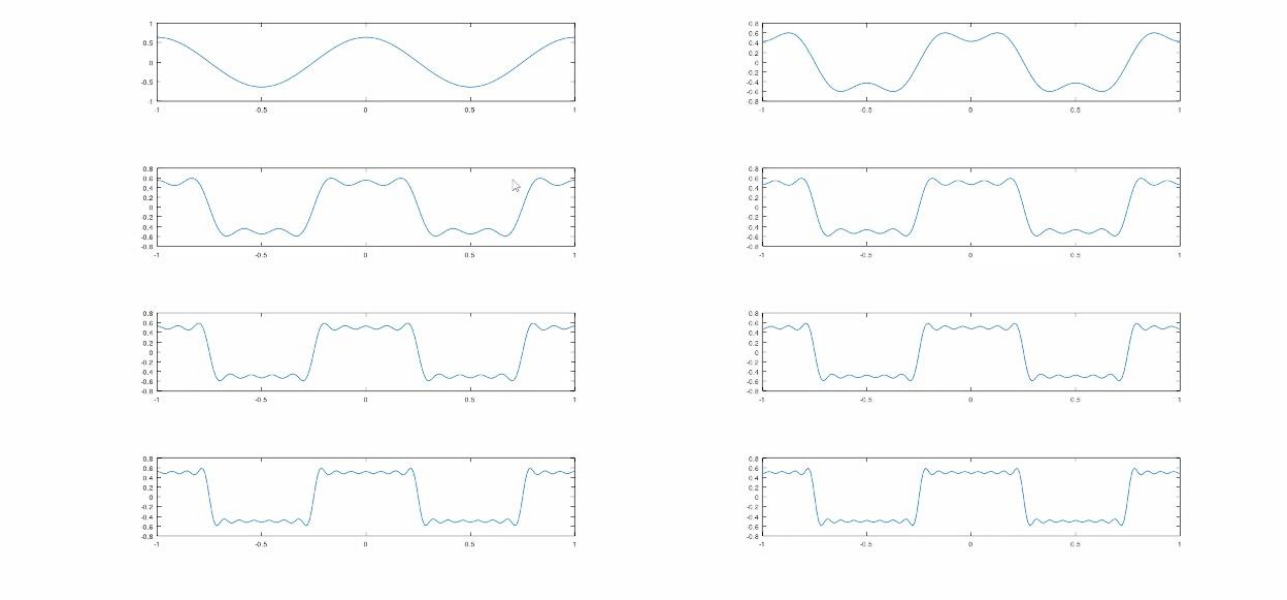
```
% Формирование массива x:
x=-10:0.1:10;
% Формирование массива y.
y1=sin(x)+1/3*sin(3*x)+1/5*sin(5*x);
y2=cos(x)+1/3*cos(3*x)+1/5*cos(5*x);
% Построение графика функции:
plot(x,y1, "-ok; y1=sin(x)+(1/3)*sin(3*x)+(1/5)*sin(5*x);","markersize",4)
hold on;
plot(x,y2, "-ok; y2=cos(x)+(1/3)*cos(3*x)+(1/5)*cos(5*x);","markersize",4)
% 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)\ny2=cos x+ (1/3)cos(3x)+(1/5)cos(5x)')
% Экспорт рисунка в файл .eps:
print ("plot-sin.eps", "-mono", "-FArial:16", "-deps")
% Экспорт рисунка в файл .png:
print("plot-sin.png");
```



4. Создаём новый сценарий и сохраняем его с именем meandr.m. Втсавляем туда код из методички, запускаем и видим графики с разными количествами гармоник

plot_sincos.m | meandr.m

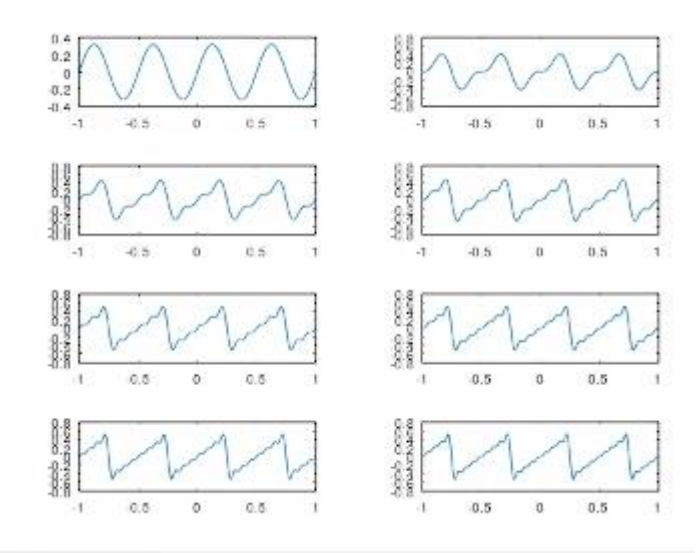
```
% meandr.m
% количество отсчетов (гармоник):
N=8;
% частота дискретизации:
t=-1:0.01:1;
% значение амплитуды:
A=1;
% период:
T=1;
% амплитуда гармоник
nh=(1:N)*2-1;
% массив коэффициентов для ряда, заданного через cos:
Am=2/pi ./ nh;
Am(2:2:end) = -Am(2:2:end);
% массив гармоник:
harmonics=cos(2 * pi * nh' * t/T);
% массив элементов ряда:
s1=harmonics.*repmat(Am',1,length(t));
% Суммирование ряда:
s2=cumsum(s1);
% Построение графиков:
for k=1:N
subplot(4,2,k)
plot(t, s2(k,:))
end
```



5. Меняем формат файла на png.Меняем формулу косинусы на синусы и запускаем программу.

plot_sincos.m | meandrsin.m | plot_sin.m

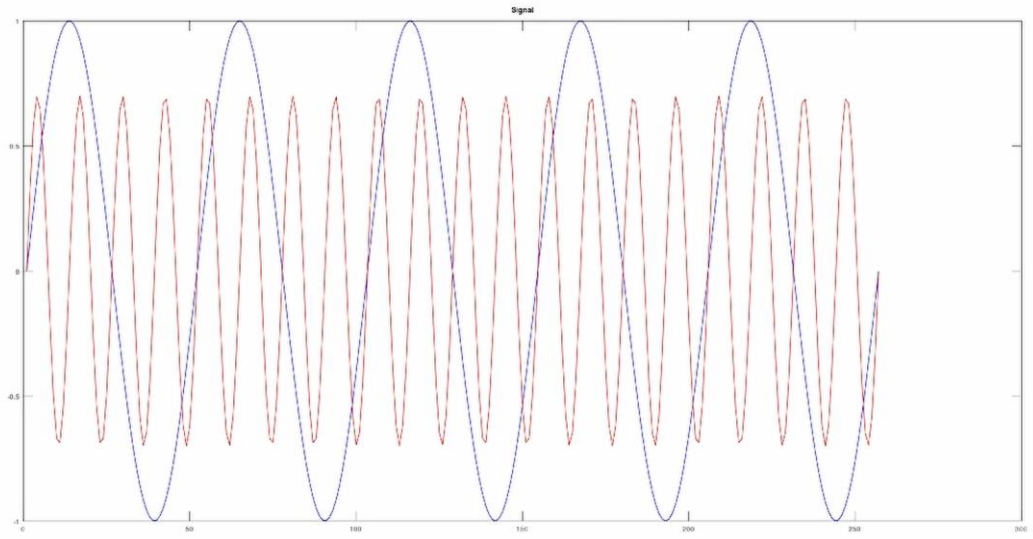
```
% meandr.m
% количество отсчетов (гармоник):
N=8;
% частота дискретизации:
t=-1:0.01:1;
% значение амплитуды:
A=1;
% период:
T=1;
% амплитуда гармоник
nh=(1:N)*2;
% массив коэффициентов для ряда, заданного через cos:
Am=2/pi ./ nh;
Am(2:2:end) = -Am(2:2:end);
% массив гармоник:
harmonics=sin(2 * pi * nh' * t/T);
% массив элементов ряда:
s1=harmonics.*repmat(Am',1,length(t));
% Суммирование ряда:
s2=cumsum(s1);
% Построение графиков:
for k=1:N
subplot(4,2,k)
plot(t, s2(k,:))
print("meandr.png")
```



6. Создаем каталог spectre1 и в нем файл spectre.m. и вводим туда код из методички

spectre.m

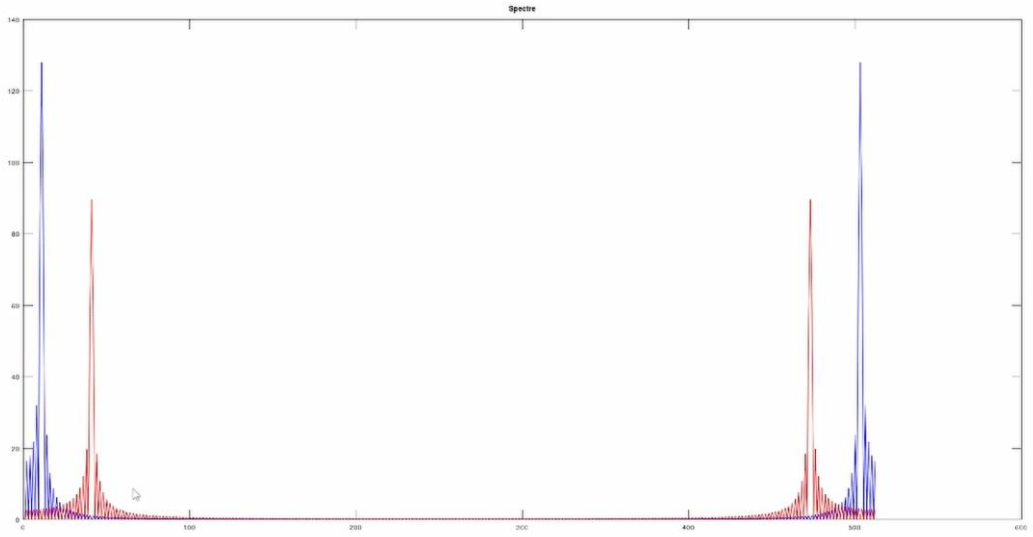
```
% spectre1/spectre.m
% Создание каталогов signal и spectre для размещения графиков:
mkdir 'signal';
mkdir 'spectre';
% Длина сигнала (с):
tmax = 0.5;
% Частота дискретизации (Гц) (количество отсчетов):
fd = 512;
% Частота первого сигнала (Гц):
f1 = 10;
% Частота второго сигнала (Гц):
f2 = 40;
% Амплитуда 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';

```



7. С помощью быстрого преобразования Фурье находим спектры сигналов добавив в файл spectre.m еще немного кода. Меняем график спектра: отбрасывем дублирующие отрицательные частоты, а также берем расчёт то, что на каждом шаге вычисления быстрого преобразования Фурье происходит суммирование амплитуд сигналов. При сравнении видно, что в исправленном графики отсутствуют отрицательные частоты, а также не суммируются амплитуды при каждом шаге

spectre.m

```
f2 = 40;
% Амплитуда первого сигнала:
a1 = 1;
% Амплитуда второго сигнала:
a2 = 0.7;
% Массив отсчётов времени:
t = 0:1./fd:tmax;
% Спектр сигнала:
fd2 = fd/2;
% Два сигнала разной частоты:
signal1 = a1*sin(2*pi*t*f1);
signal2 = a2*sin(2*pi*t*f2);
% График 1-го сигнала:
plot(signal1,'b');
% График 2-го сигнала:
hold on
plot(signal2,'r');
hold off
title('Signal');
% Экспорт графика в файл в каталоге signal:
print 'signal/spectre.png';
% Посчитаем спектр
% Амплитуды преобразования Фурье сигнала 1:
spectre1 = abs(fft(signal1,fd));
% Амплитуды преобразования Фурье сигнала 2:
spectre2 = abs(fft(signal2,fd));
% Построение графиков спектров сигналов:
plot(spectre1,'b');
hold on
plot(spectre2,'r');
hold off
title('Spectre');
print 'spectre/spectre.png';
```

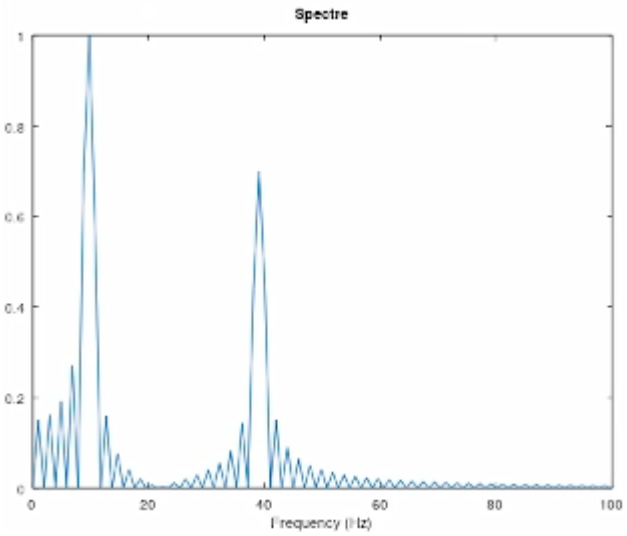


8. Находим спектр суммы рассмотренных сигналов, создав каталог spectr_sum и файл в нем spectre_sum.m.

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 отсчетов):
fd = 512;
% Частота первого сигнала (Гц):
f1 = 10;
% Частота второго сигнала (Гц):
f2 = 40;
% Амплитуда первого сигнала:
a1 = 1;
% Амплитуда второго сигнала:
a2 = 0.7;
% Спектр сигнала
fd2 = fd/2;
% Сумма двух сигналов (синусоиды) разной частоты:
% Массив отсчётов времени:
t = 0:1./fd:tmax;
signal1 = a1*sin(2*pi*t*f1);
signal2 = a2*sin(2*pi*t*f2);
signal = signal1 + signal2;
plot(signal);
title('Signal');
print 'signal/spectre_sum.png';
% Подсчет спектра:
% Амплитуда 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;
```

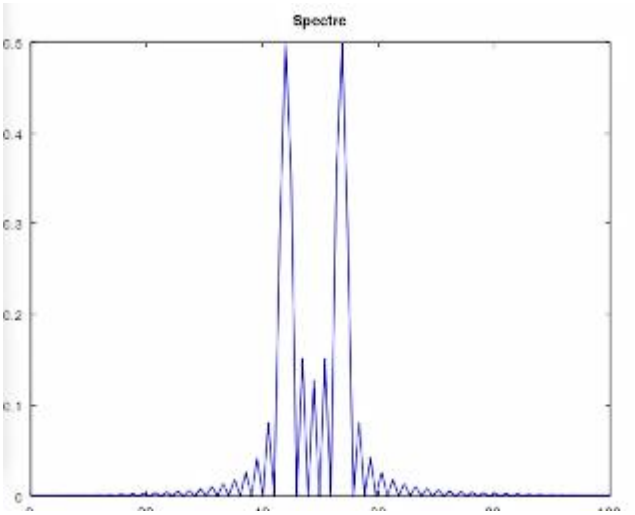
строка: 17 столбец: 13 кодировка: UTF-8 конец строки: CRLF



9. Создаем каталог modulation и в нем файл am.m вводим туда код из методички, запускаем программу и видим, что спектр произведения представляет собой свёртку спектров.

am.m

```
% modulation/am.m
% Создание каталогов signal и spectre для размещения графиков:
mkdir 'signal';
mkdir 'spectre';
% Модуляция синусоид с частотами 50 и 5
% Длина сигнала (с)
tmax = 0.5;
% Частота дискретизации (Гц) (количество отсчётов)
fd = 512;
% Частота сигнала (Гц)
f1 = 5;
% Частота несущей (Гц)
f2 = 50;
% Спектр сигнала
fd2 = fd/2;
% Построение графиков двух сигналов (синусоиды)
% разной частоты
% Массив отсчётов времени:
t = 0:1./fd:tmax;
signal1 = sin(2*pi*t*f1);
signal2 = sin(2*pi*t*f2);
signal = signal1 .* signal2;
plot(signal, 'b');
hold on
% Построение огибающей:
plot(signal1, 'r');
plot(-signal1, 'r');
hold off
title('Signal');
print 'signal/am.png';
% Расчет спектра:
% Амплитуды преобразования Фурье-сигнала:
spectre = fft(signal,fd);
% Сетка частот:
```



10. Проверяем установлен ли у меня signal.

Создаем каталог coding и в нем создаем файлы main.m, maptowave.m, unipolar.m, ami.m, bipolarnrz.m, bipolarrz.m, manchester.m, diffmanc.m, calcspectre.m и прописываем нужный код из методички

main.m | maptowave.m | unipolar.m | ami.m | bipolarnrz.m | bipolarrz.m | manchester

```
70  title('Manchester');
71  print 'sync/manchester.png';
72  % Дифференциальное манчестерское кодирование
73  wave=diffmanc(data_sync);
74  plot(wave)
75  title('Differential Manchester');
76  print 'sync/diffmanc.png';
77  % Униполярное кодирование:
78  wave=unipolar(data_spectre);
79  spectre=calcspectre(wave);
80  title('Unipolar');
81  print 'spectre/unipolar.png';
82  % Кодирование AMI:
83  wave=ami(data_spectre);
84  spectre=calcspectre(wave);
85  title('AMI');
86  print 'spectre/ami.png';
87  % Кодирование NRZ:
88  wave=bipolarnrz(data_spectre);
89  spectre=calcspectre(wave);
90  title('Bipolar Non-Return to Zero');
91  print 'spectre/bipolarnrz.png';
92  % Кодирование RZ:
93  wave=bipolarrz(data_spectre);
94  spectre=calcspectre(wave);
95  title('Bipolar Return to Zero');
96  print 'spectre/bipolarrz.png';
97  % Манчестерское кодирование:
98  wave=manchester(data_spectre);
99  spectre=calcspectre(wave);
100 title('Manchester');
101 print 'spectre/manchester.png';
102 % Дифференциальное манчестерское кодирование:
```

строка: 86 | столбец: 1 | кодировка: UTF-8 | конец строки: CRLF

Вывод:

Мы изучили методы кодирования и модуляции сиогналов с помощью языка octave. Научились определять параметры сигнала. Изучили основы принципов модуляции на примере аналоговой амплитудной модуляции.