

Лабораторная работа №1

Методы кодирования и модуляция сигналов

Лисовская А.В.

18 декабря 2025

Российский университет дружбы народов, Москва, Россия

Докладчик

- Лисовская Арина Валерьевна
- Студент, НПИбд01-23
- Российский университет дружбы народов
- 1132236814@rudn.ru

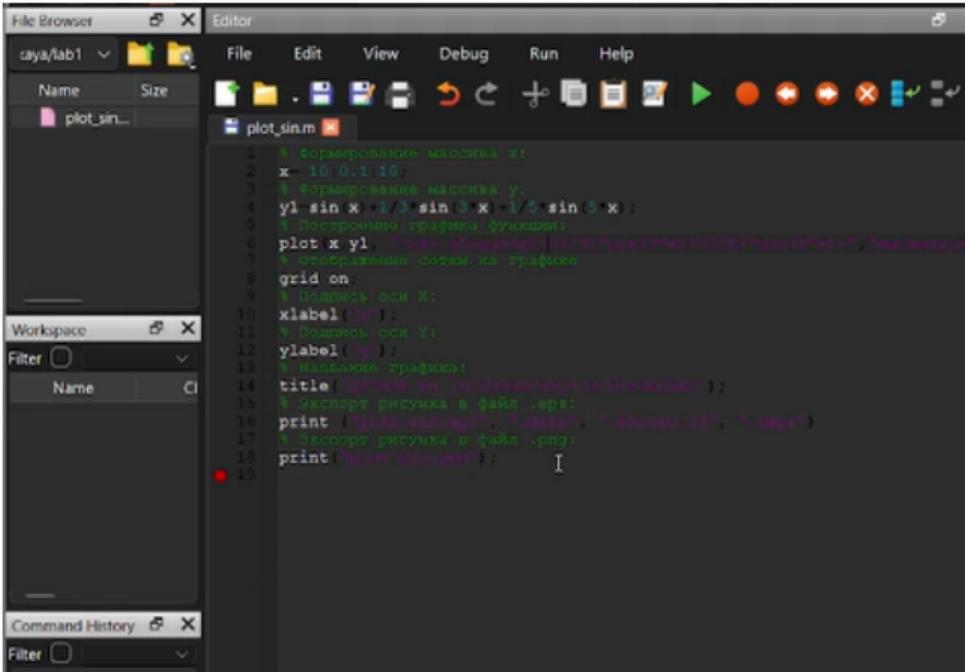
Цель работы

- Изучение методов кодирования и модуляции сигналов в Octave.
- Определение спектральных характеристик сигналов.
- Демонстрация принципов модуляции сигнала на примере аналоговой амплитудной модуляции.
- Исследование свойства самосинхронизации различных кодов.

Часть 1: Построение графиков и ряды Фурье

Графики функций y_1 и y_2

На одном графике представлены две сложные функции (суммы гармоник синуса и косинуса). Демонстрируется работа с `plot()` и настройка стилей линий.



The screenshot shows the MATLAB interface with the following components:

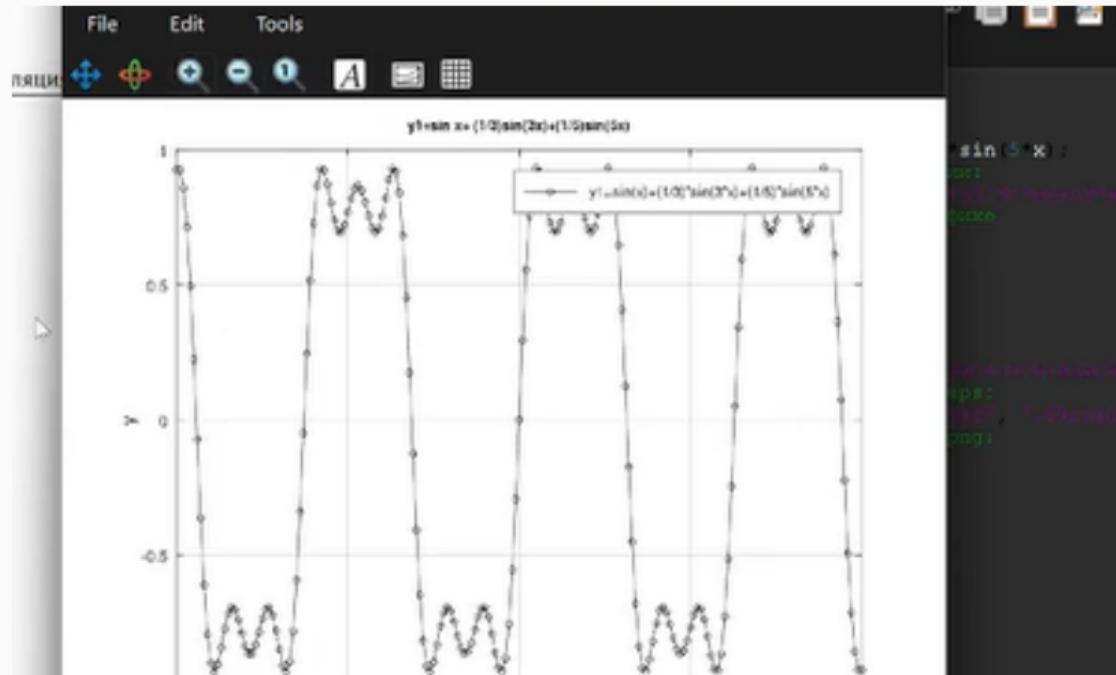
- File Browser:** Shows the current folder as "daya/lab1".
- Editor:** Displays the script file "plot_sin.m" with the following code:

```
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 plot(x, y1, 'r', 'LineWidth', 2, 'Marker', 'none', 'LineStyle', 'solid');
7 % Отображение сетки на графике
8 grid on;
9 % Подпись оси X:
10 xlabel('x');
11 % Подпись оси Y:
12 ylabel('y');
13 % Назование графика:
14 title('График двух гармоник');
15 % Экспорт рисунка в файл .eps:
16 print('График двух гармоник', 'EPS', 'Figure', 'Figure');
17 % Экспорт рисунка в файл .png:
18 print('График двух гармоник', 'png');
```

- Workspace:** Shows the variable "x" defined as a double array.
- Command History:** Shows the command history.

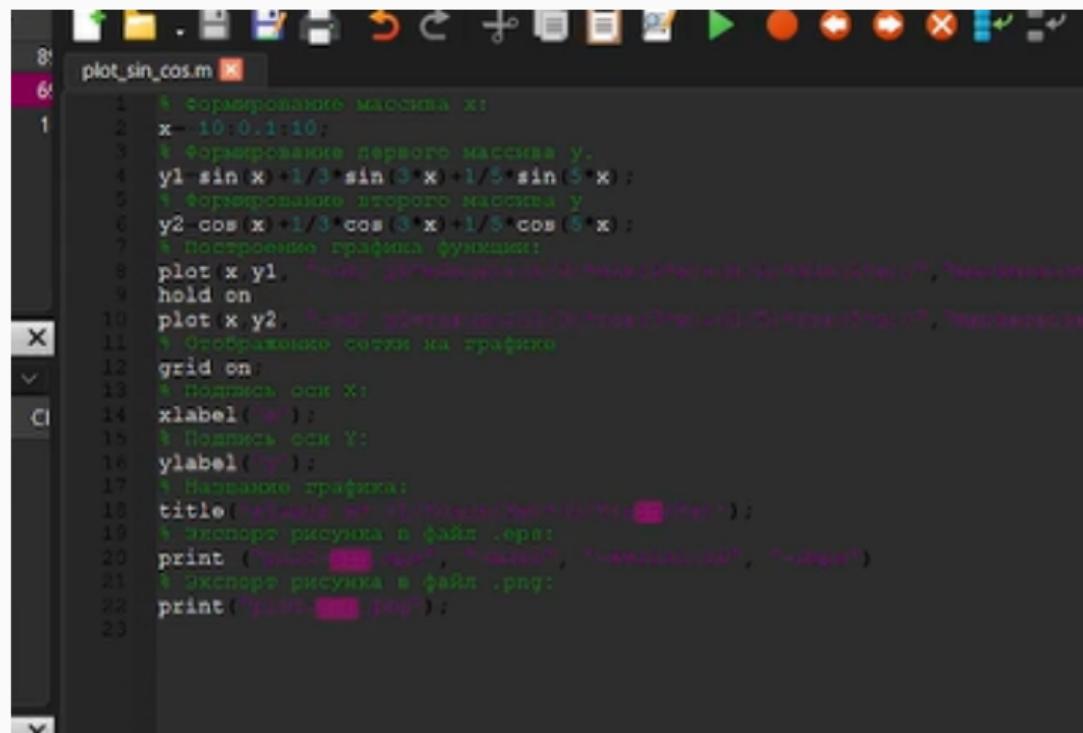
Разложение меандра: Код meandr.m

Фрагмент кода, реализующий разложение меандра в ряд Фурье через косинусы. Задаются параметры гармоник N, время t, амплитуда A и период T.



Разложение меандра: Код meandr_sin.m

Альтернативный вариант реализации меандра через синусы. Показывает понимание двух форм ряда Фурье для одной и той же функции.

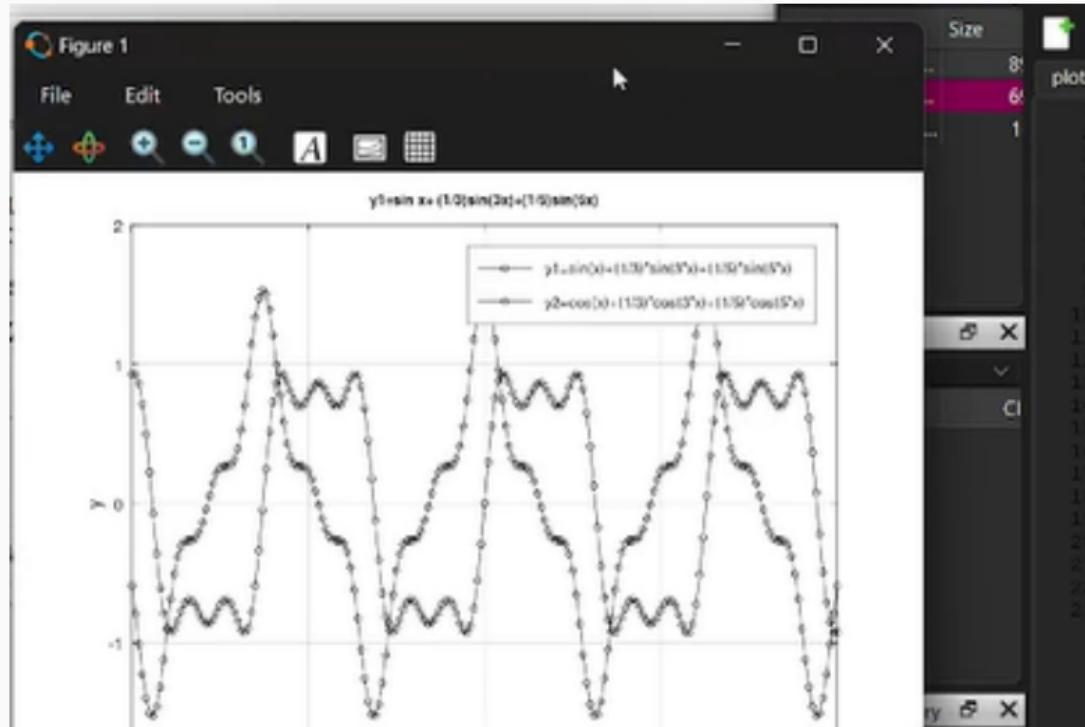


```
plot_sin_cos.m
1 % формирование массива x:
2 x= 10:0.1:10;
3 % формирование первого массива у.
4 y1 sin x +1/3*sin(3*x)+1/5*sin(5*x);
5 % формирование второго массива у
6 y2 cos x +1/3*cos(3*x)+1/5*cos(5*x);
7 % построение графика функций:
8 plot x y1; hold on
9 plot x y2; hold on
10 % Отображение сетки на графике
11 grid on
12 % Подпись оси x:
13 xlabel('x');
14 % Подпись оси Y:
15 ylabel('y');
16 % Назначение графика:
17 title('График двух синусоид');
18 % экспорт рисунка в файл jpg:
19 print ('График двух синусоид.jpg','jpg');
20 % Экспорт рисунка в файл png:
21 print ('График двух синусоид.png','png');
```

Часть 2: Спектральный анализ (БПФ)

Подготовка к анализу: Код spectre.m

Создание двух синусоидальных сигналов разной частоты. Код задаёт параметры (fd , $f1$, $f2$, $a1$, $a2$) и формирует массивы.



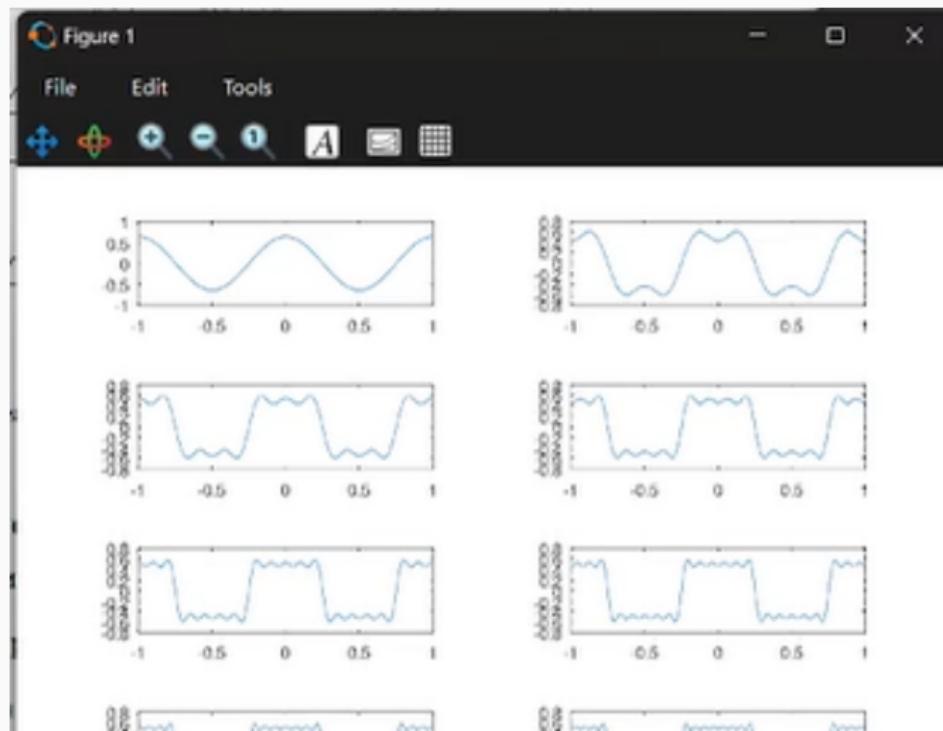
Исходные сигналы во временной области

Визуализация двух сигналов: $\sin(2\pi \cdot 10 \cdot t)$ и $0.7 \cdot \sin(2\pi \cdot 40 \cdot t)$.

```
plot_sin_cos.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 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 % экспорт рисунка в файл .eps:
27 print('meandr.eps','-r1000','/EPS/1000','/Landscape')
28 % экспорт рисунка в файл .png:
29 print('meandr.png');
```

Необработанный спектр сигналов

График амплитудного спектра без коррекции. Демонстрирует исходный результат применения БПФ с дублирующимися частотами.



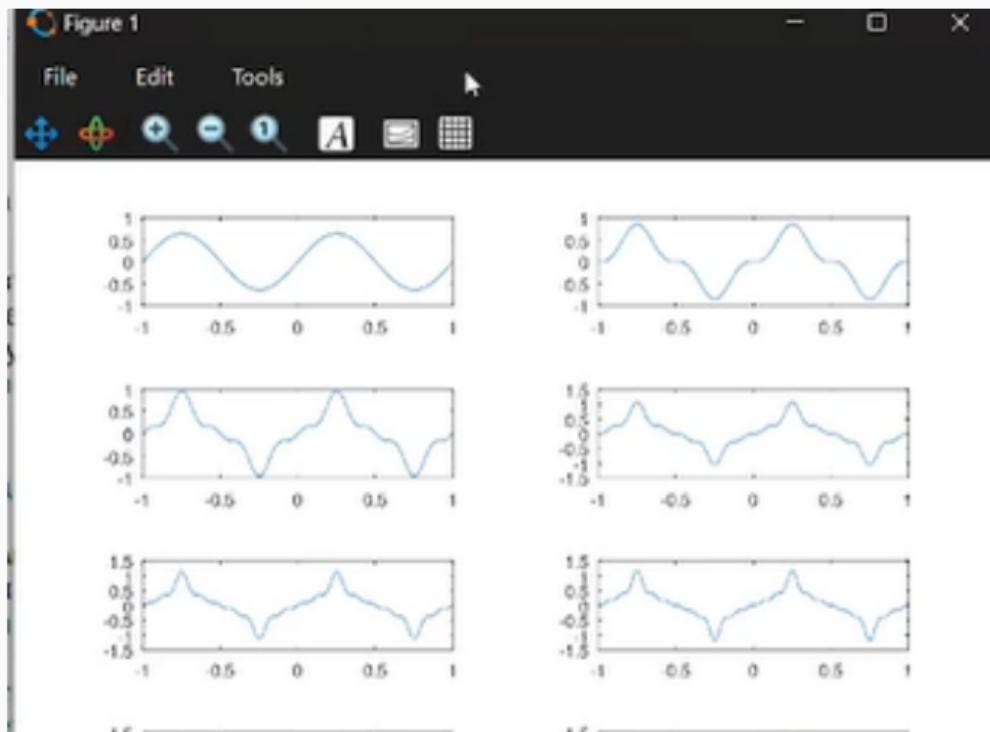
Исправленный спектр сигналов

Нормированный спектр с отброшенными отрицательными частотами. Четко видны пики на частотах 10 Гц и 40 Гц.

```
meandr.sin.m x spectrem.m x
1 % meandr.sin.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*pi;
12 % массив коэффициентов для ряда, заданного через sin:
13 Am=2/pi.*nh;
14 Am(2:2:nd)=Am(2:2:nd);
15 % массив гармоник:
16 harmonics=sin(2*pi.*nh.*t);
17 % массив суммировавшего ряда:
18 s1=harmonics.*harmonics.*length(t);
19 % суммирование ряда:
20 s2=cumsum(s1);
21 % Воспроизведение графиков:
22 for k=1:N
23 subplot(4,2,k)
24 plot(t,s2(k,:));
25 end
26 % Экспорт рисунка в файл jpeg:
27 print('meandr-sin.jpeg','jpeg','/home/alexey','/Desktop')
28 % Экспорт рисунка в файл eps:
29 print('meandr-sin.eps','eps','/home/alexey','/Desktop')
```

Код расчёта суммы: spectre_sum.m

Код для проверки свойства аддитивности: спектр суммы сигналов должен быть равен сумме спектров.



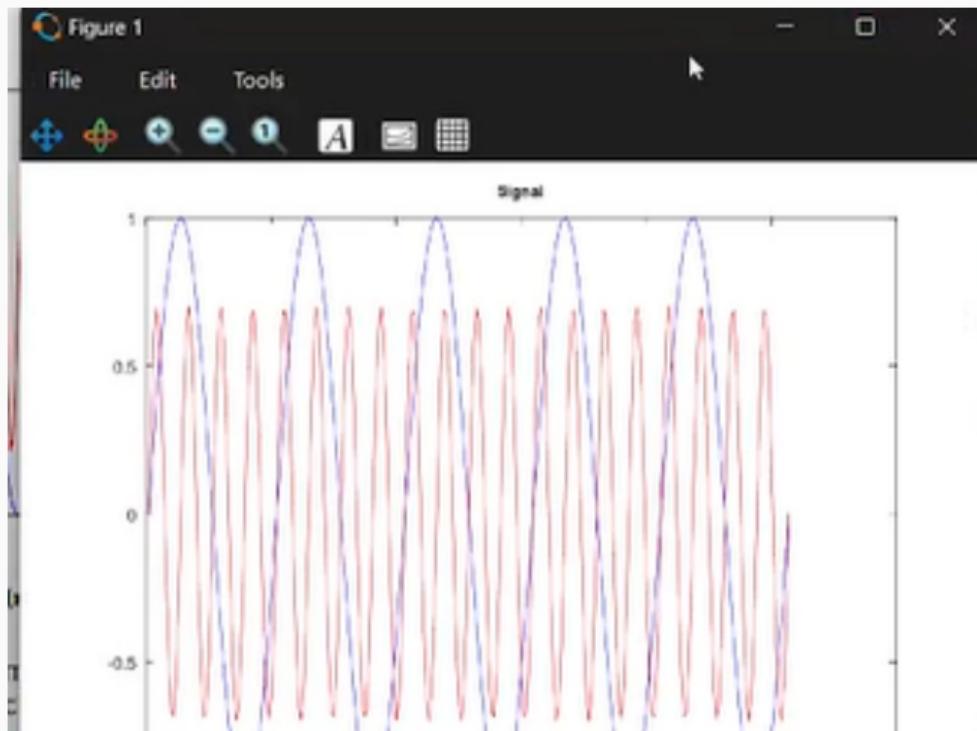
Спектр суммарного сигнала

Подтверждение линейности преобразования Фурье. Спектр содержит те же частотные компоненты (10 и 40 Гц).

```
% spectrel/spectre.m
% Создание каталогов signal и spectra для размещения графиков;
mkdir signal;
mkdir spectra;
% Длительность 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 отсчетов времени:
t = 0:1./fd:tmax;
% Сигнал 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, '-');
% График 2-го сигнала:
hold on
plot(signal2, '--');
% График суммарного сигнала:
plot(signal1+signal2, 'r');
% График спектра суммарного сигнала:
figure(2);
fft = abs(fftshift(fft(signal1+signal2)));
% График спектра первого сигнала:
figure(3);
fft1 = abs(fftshift(fft(signal1)));
% График спектра второго сигнала:
figure(4);
fft2 = abs(fftshift(fft(signal2)));
```

График суммарного сигнала

Визуализация суммы сигналов во временной области перед проведением спектрального анализа.



Часть 3: Амплитудная модуляция (AM)

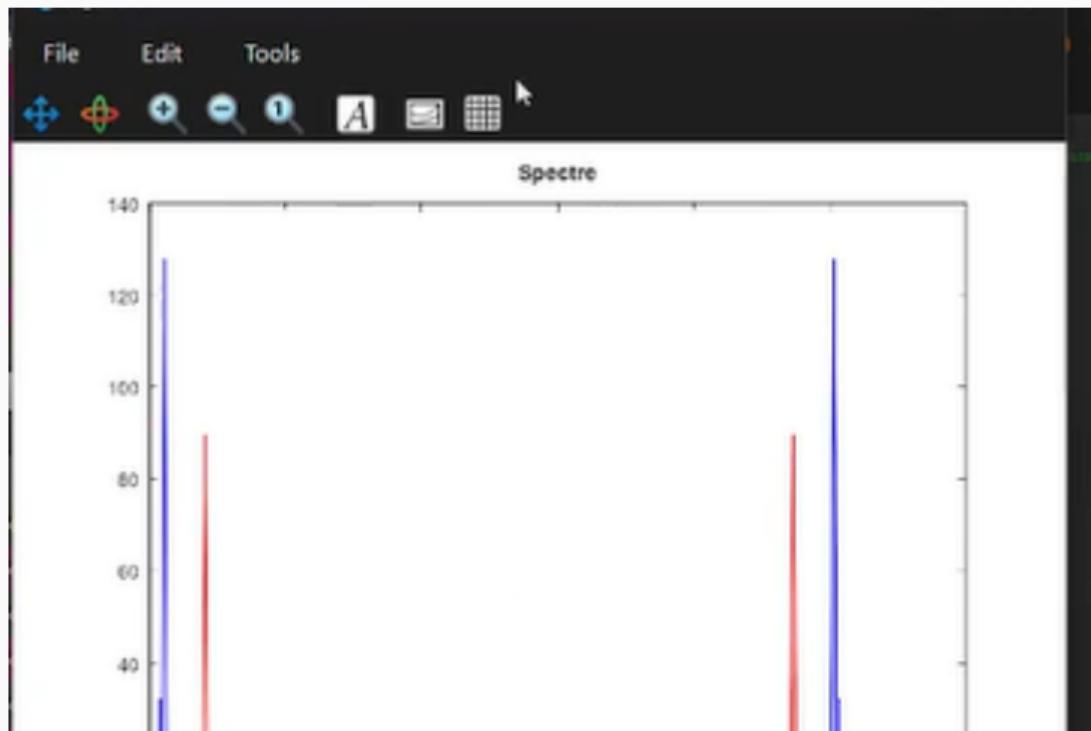
Моделирование АМ: Код am.m

Демонстрация принципов АМ. Код показывает перемножение низкочастотного сигнала (5 Гц) и несущей (50 Гц).

```
% am.m
% спектр/спектр.m
% Создание каталогов signal и spectra для размещения графиков:
mkdir 'signal';
mkdir 'spectra';
% Длительность сигнала (с):
tmax = 0.5;
% Частота дискретизации (Гц) (количество отсчетов):
fd = 512;
% Частота первого сигнала (Гц):
f1 = 10;
% Частота второго сигнала (Гц):
f2 = 10;
% Амплитуда первого сигнала:
a1 = 1;
% Амплитуда второго сигнала:
a2 = 0.7;
% Массив отсчетов времени:
t = 0:1./fd:tmax;
% Сгенерир. сигналы:
fd2 = fd/2;
% Два сигнала разной частоты:
signal1 = a1*sin(2*pi*t*f1);
signal2 = a2*sin(2*pi*t*f2);
% Посчитаем спектр
% Амплитуда преобразования Fourier сигнала 1:
spectrel = abs(fft(signal1,fd));
% Амплитуда преобразования Fourier сигнала 2:
spectre2 = abs(fft(signal2,fd));
% Достроение графиков спектров сигналов:
plot(spectrel(1:fd2))
```

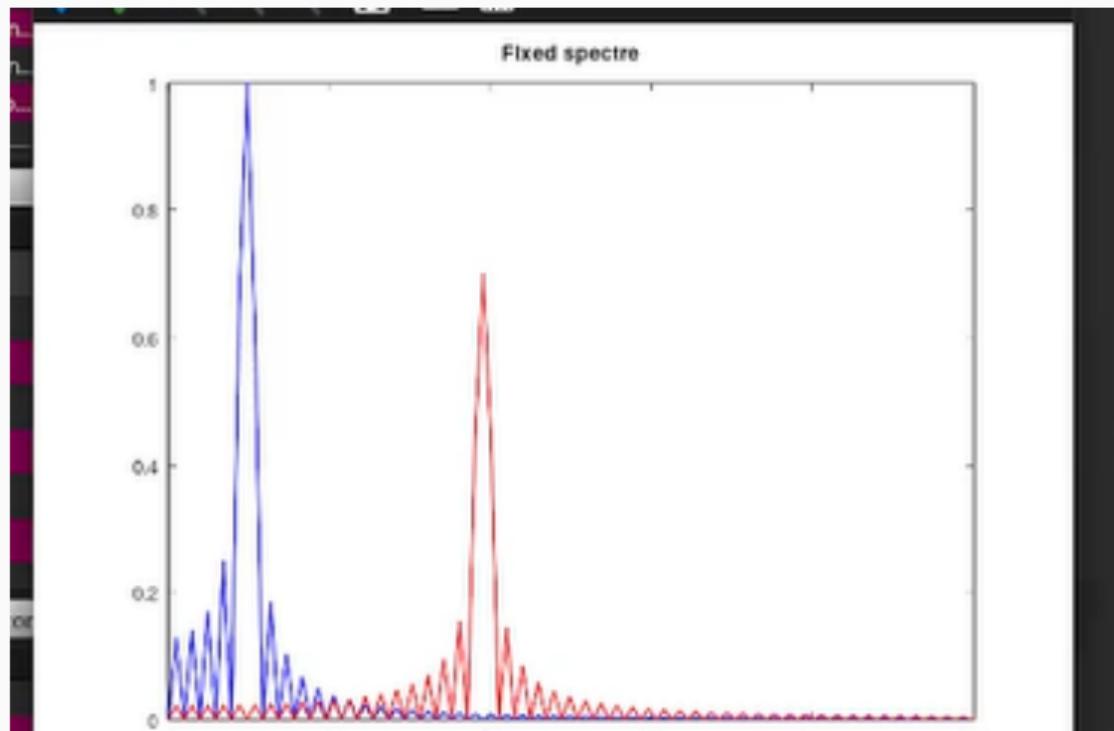
Спектр АМ-сигнала

Иллюстрация свойства модуляции: спектр содержит несущую (50 Гц) и боковые полосы (45 Гц и 55 Гц).



АМ-сигнал и его огибающая

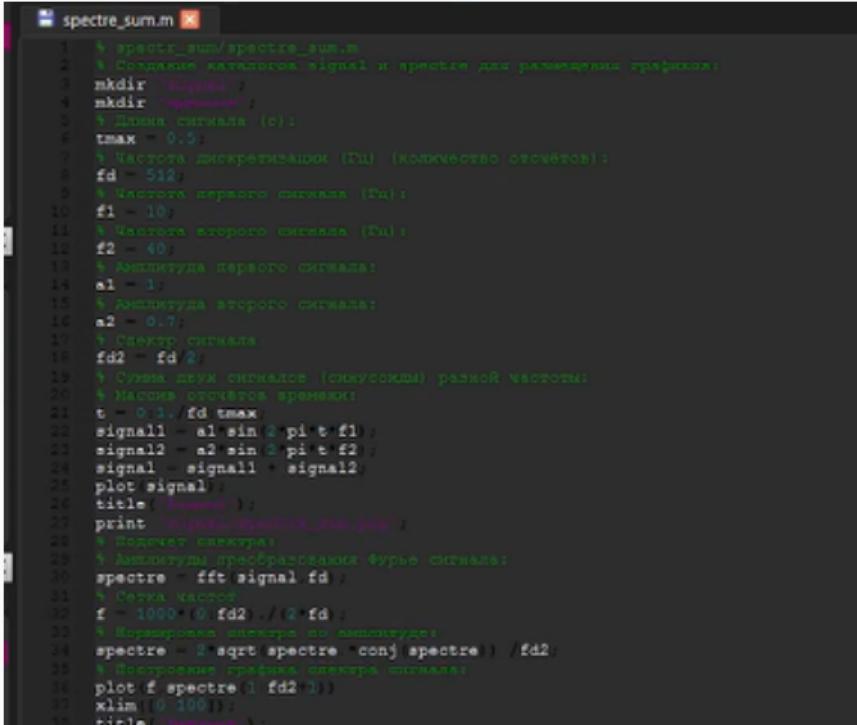
Наглядно показано, как изменяется амплитуда несущей в соответствии с модулирующим сигналом.



Часть 4: Методы цифрового кодирования

Установленные пакеты Octave

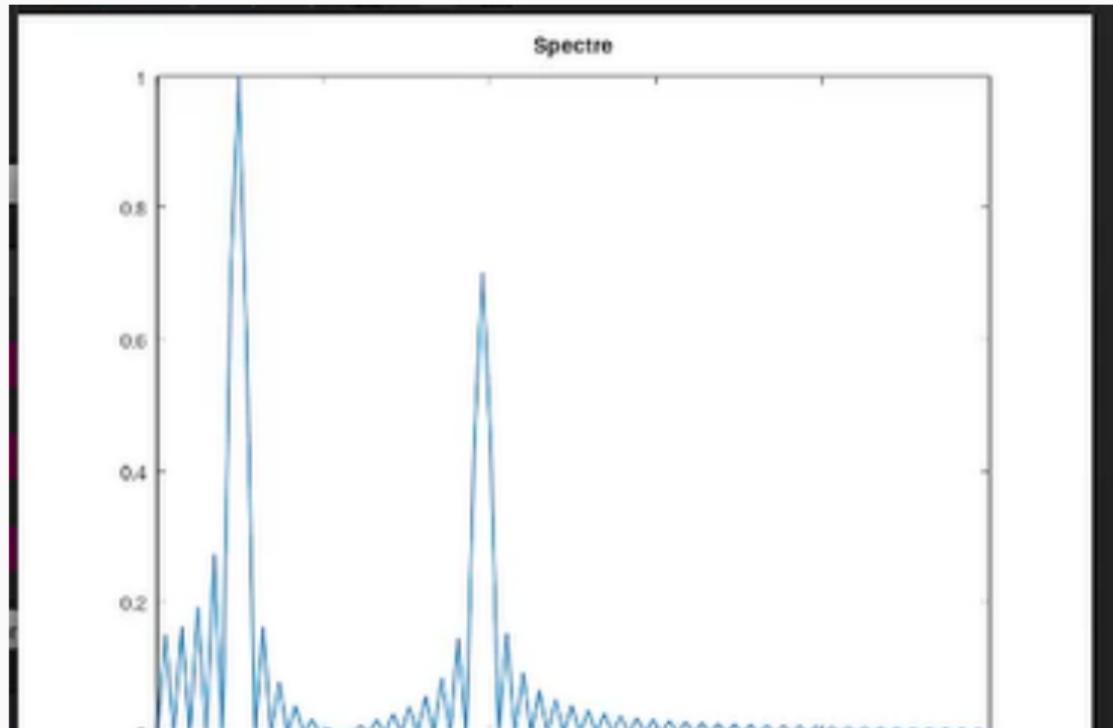
Проверка наличия пакета `signal`, необходимого для корректной работы функций кодирования.



```
1 % spectre_sum/spectre_sum.m
2 % Создание каталогов signal и spectre для размещения графиков:
3 mkdir 'spectre';
4 mkdir 'signal';
5 % Ширина сигнала (с):
6 tmax = 0.5;
7 % Частота дискретизации (ЦЦ) (количество отсчетов):
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 % Спектр сигнала
18 fd2 = fd/2;
19 % Сумма двух сигналов (синусоиды) разной частоты
20 % Насоки отмечаются временем:
21 t = 0:1./fd:tmax;
22 signal1 = a1*sin(2*pi*t*f1);
23 signal2 = a2*sin(2*pi*t*f2);
24 signal = signal1 + signal2;
25 plot(signal);
26 title('Сигнал');
27 print('Сигнал-сумма.jpg');
28 % Помогает спектру:
29 % Амплитуда преобразования Фурье сигнала:
30 spectre = fft(signal,fd);
31 % Секва частот
32 f = 1000*(0:fd2)./(2*fd);
33 % Нормирован спектра по амплитуде:
34 spectre = 2*sqrt(spectre.*conj(spectre))./fd2;
35 % Накрашены красным спектра сигналов:
36 plot(f,spectre);
37 xlim([0,100]);
38 title('Спектр');
```

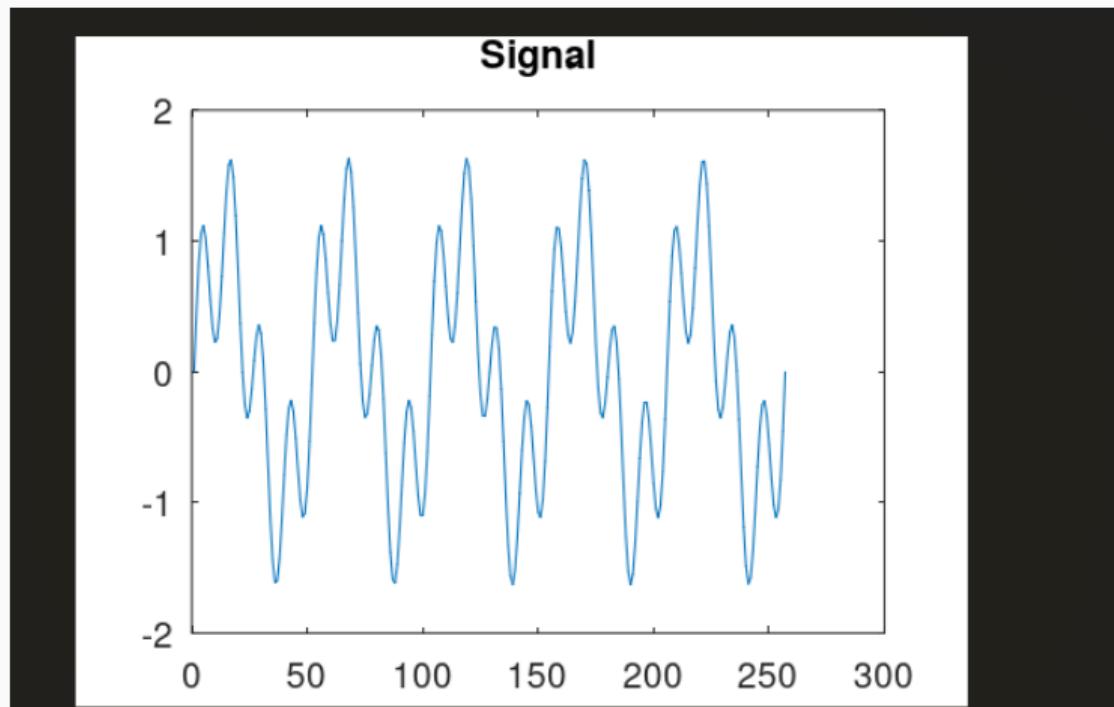
Главный скрипт: main.m

Инициализация данных, создание каталогов и вызов функций кодирования.
Основа для построения графиков.



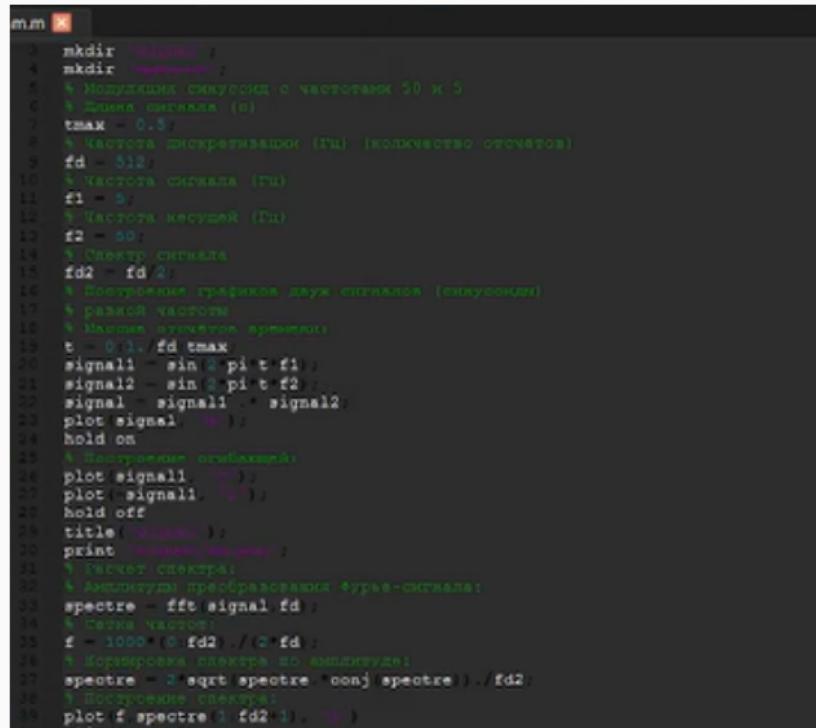
Функция `maptowave.m`

Вспомогательная функция для преобразования битовой последовательности в волновой сигнал для визуализации.



Функция кодирования AMI

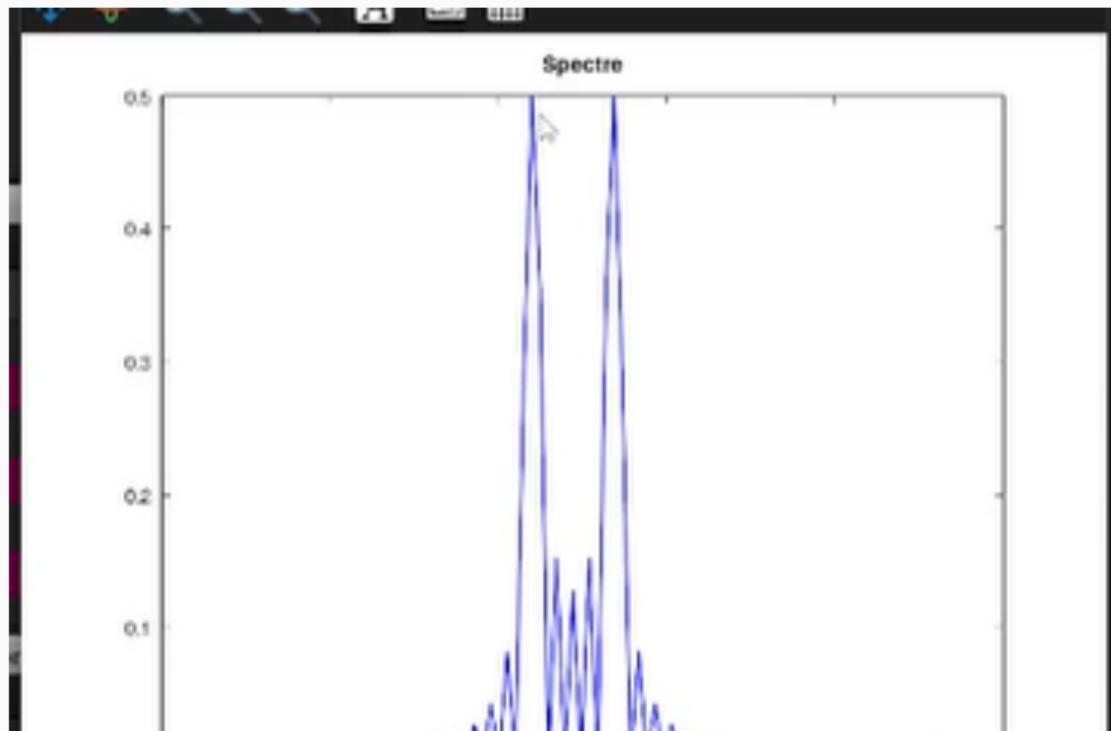
Реализация метода Alternate Mark Inversion. Показывает чередование полярности для логических единиц.



```
m.m
1 mkdir 'AMI';
2 mkdir 'spectrum';
3 % Модуляция синусом с частотами 50 и 5
4 % Длительность сигнала (с)
5 tmax = 0.5;
6 % Частота дискретизации (Гц) (количество отсчетов)
7 fd = 512;
8 % Частота сигнала (Гц)
9 f1 = 5;
10 % Частота кесущей (Гц)
11 f2 = 50;
12 % Спектр сигнала
13 fd2 = fd/2;
14 % Построение графиков двух сигналов (синусовидн.)
15 % разной частоты
16 % Максимум отсчетов времени
17 t = 0 : 1 / fd : tmax;
18 signal1 = sin(2 * pi * t * f1);
19 signal2 = sin(2 * pi * t * f2);
20 signal = signal1 .* signal2;
21 plot(signal, 'r');
22 hold on
23 % Построение огибающей
24 plot(signal, 'g');
25 plot(signal1, 'b');
26 plot(signal2, 'm');
27 hold off
28 title('AMI');
29 print 'AMI.mpl';
30 % Спектр сигнала
31 % Амплитуда преобразованного Фурье-сигнала:
32 spectre = fft(signal, fd);
33 % Стока частот:
34 f = 1000 * (0 : fd2) ./ (2 * fd);
35 % Корректировка спектра по амплитуде:
36 spectre = 2 * sqrt(spectre * conj(spectre)) ./ fd2;
37 % Построение спектра:
38 plot(f, spectre(1 : fd2 / 2), 'r')
```

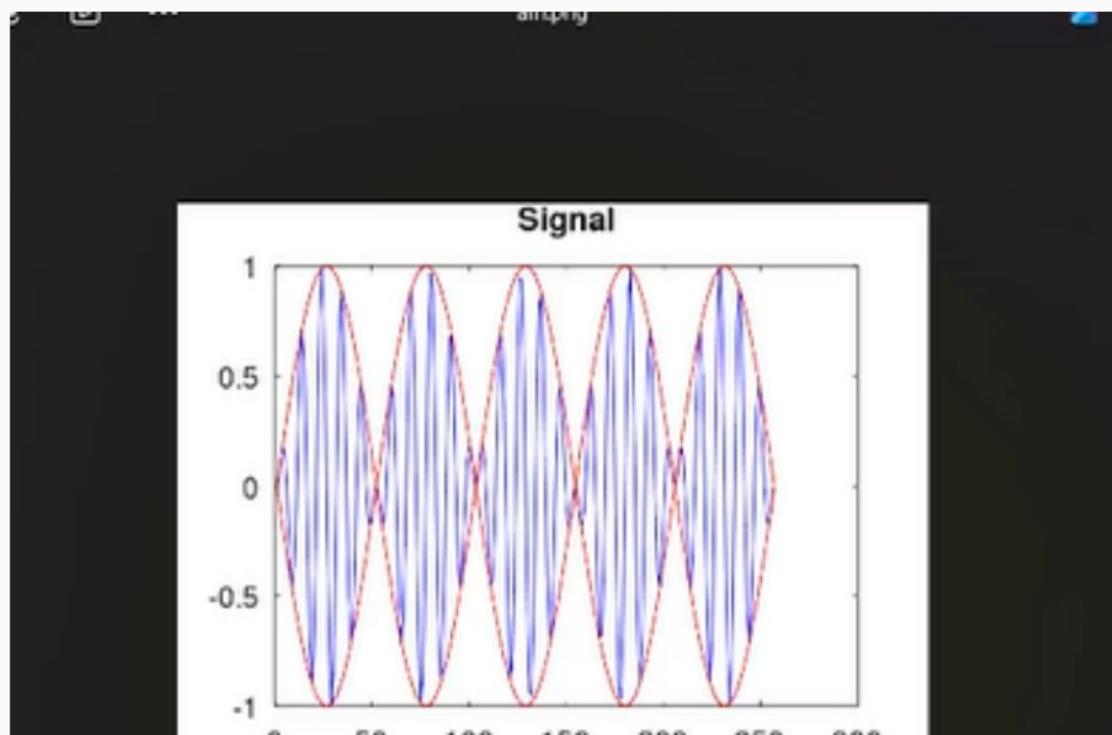
Функция кодирования RZ

Реализация метода Return to Zero (возврат к нулю). Обеспечивает переход в середине каждого бита.



Функция Manchester

Реализация кода с обязательным переходом в центре каждого битового интервала для самосинхронизации.



Функция Diff. Manchester

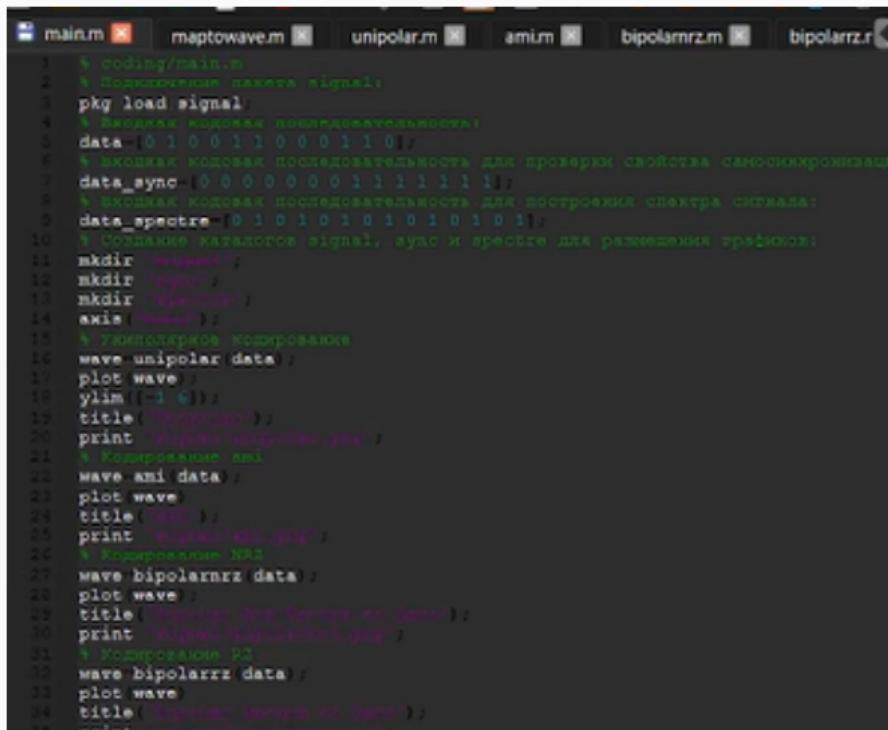
Более сложный метод, где информация кодируется наличием или отсутствием перепадов уровня.

>> pkg list	Package Name	Version	Installation directory
	audio	2.0.9	...\octave\packages\audio-2.0.9
	biosig	3.9.0	...\octave\packages\biosig-3.9.0
	cfitsio	0.0.7	...\octave\packages\cfitsio-0.0.7
	coder	1.10.1	...\octave\packages\coder-1.10.1
	communications	1.2.7	...\packages\communications-1.2.7
	control	4.1.3	...\octave\packages\control-4.1.3
	data-smoothing	1.3.0	...\packages\data-smoothing-1.3.0
	database	2.4.4	...\octave\packages\database-2.4.4
	dataframe	1.2.0	...\octave\packages\dataframe-1.2.0
	dicom	0.6.1	...\octave\packages\dicom-0.6.1
	financial	0.5.4	...\octave\packages\financial-0.5.4
	fl-core	1.0.2	...\octave\packages\fl-core-1.0.2
	fuzzy-logic-toolkit	0.6.2	...\fuzzy-logic-toolkit-0.6.2
	ga	0.10.4	...\share\octave\packages\ga-0.10.4

Рис. 20: Код diffmanchester.m

График кодирования АМI

Пример кодирования последовательности. Демонстрация отсутствия постоянной составляющей за счет смены полярности.

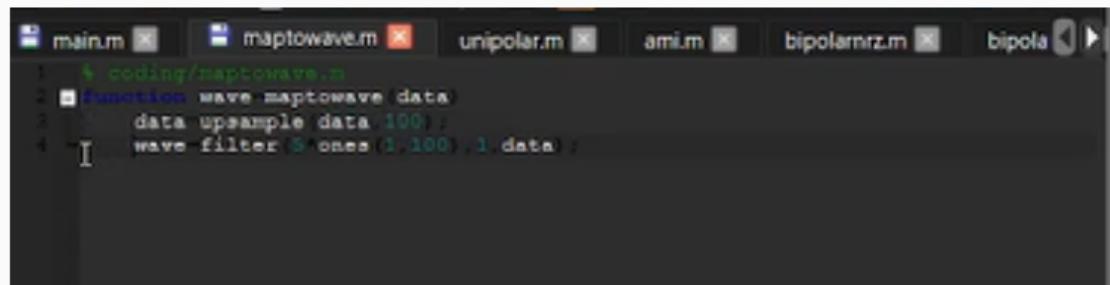


The screenshot shows a MATLAB interface with several tabs open. The tabs include: main.m, maptowave.m, unipolar.m, am.i, bipolarnrz.m, and bipolarrz.r. The main.m file is the active tab, displaying the following code:

```
1 % coding/main.m
2 % Подготовка исходных сигналов:
3 pkg load signal;
4 % Амплитуда кодовых последовательностей:
5 data=[0 1 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 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];
10 % Создание каталогов signal, wave и wave2 для размещения графиков:
11 mkdir('signal');
12 mkdir('wave');
13 mkdir('wave2');
14 axis([0 1000 0 1]);
15 % Униполярное кодирование:
16 wave=unipolar(data);
17 plot(wave);
18 ylim([-1 1]);
19 title('Униполярный сигнал');
20 print('Униполярный сигнал.jpg');
21 % Кодирование AMI:
22 wave=ami(data);
23 plot(wave);
24 title('AMI');
25 print('Кодированный AMI.jpg');
26 % Кодирование NRZ:
27 wave=bipolarnrz(data);
28 plot(wave);
29 title('Кодированный NRZ');
30 print('Кодированный NRZ.jpg');
31 % Кодирование RZ:
32 wave=bipolarrz(data);
33 plot(wave);
34 title('Кодированный RZ');
```

График кодирования NRZ

Демонстрация простейшего двухуровневого кода без возврата к нулю (Non-Return to Zero).

A screenshot of a MATLAB interface showing several open files in the workspace. The file 'main.m' is active and contains the following code:

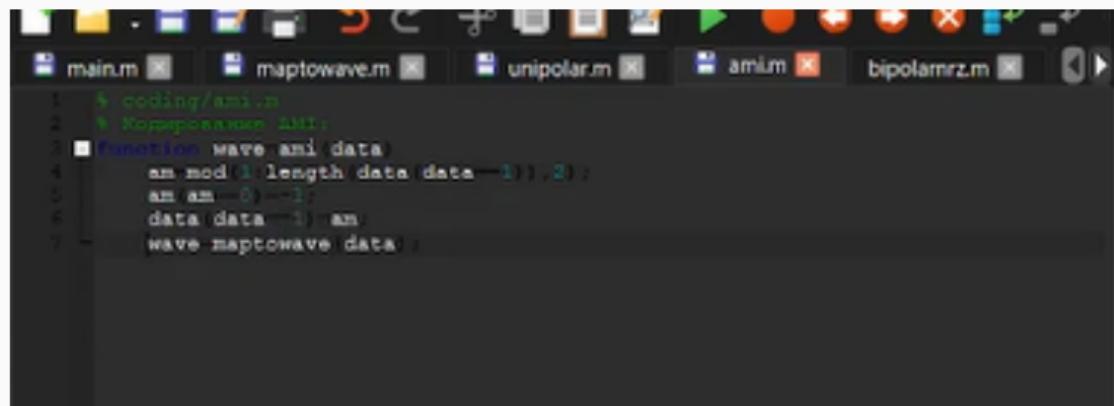
```
1 % coding/maptowave.m
2 function wave = maptowave(data)
3     data = upsample(data, 100);
4     wave = filter(ones(1, 100), 1, data);
```

The tabs for other files like 'maptowave.m', 'unipolar.m', 'ami.m', 'bipolarnrz.m', and 'bipola.m' are visible at the top.

Рис. 22: Сигнал NRZ

Дифференциальный манчестерский код

Иллюстрация метода, где перепад в начале интервала означает «0», а его отсутствие — «1».

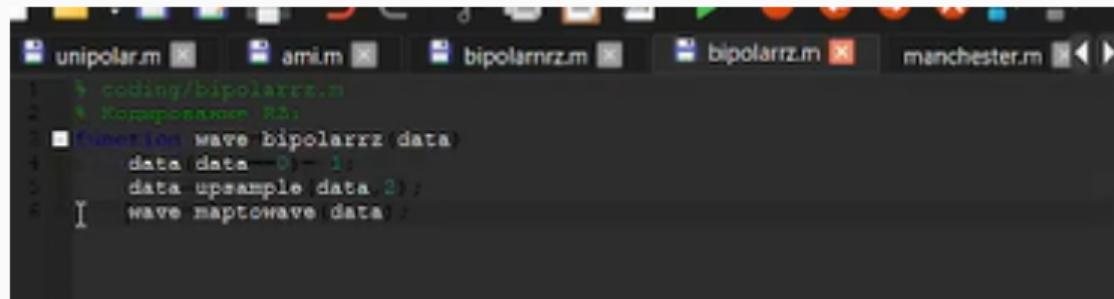


```
main.m maptowave.m unipolar.m am1.m bipolamrz.m
1 % coding/am1.m
2 % Кодирование АМ1:
3 function wave am1(data)
4     am = mod(1:length(data)-1), 2);
5     am(am==0)=1;
6     data(data==1)=am;
7     wave = maptowave(data);
```

Рис. 23: Сигнал Diff. Manchester

Манчестерское кодирование

Визуализация переходов: для «1» — переход вниз, для «0» — переход вверх в центре бита.



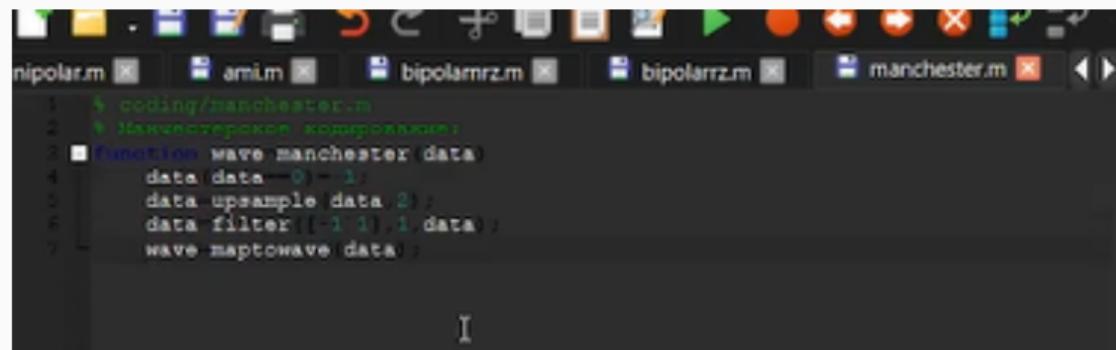
The screenshot shows a MATLAB interface with several tabs at the top: 'unipolar.m', 'am1.m', 'bipolarrrz.m', 'bipolarzz.m', and 'manchester.m'. The 'manchester.m' tab is active. Below the tabs, there is a code editor window containing the following MATLAB code:

```
1 % coding/bipolarrrz.m
2 % Компьютер РД:
3 function wave bipolarrrz(data)
4     data = data - 1;
5     data = upsample(data, 2);
6     wave = maptowave(data);
```

Рис. 24: Сигнал Manchester

Детализация формы сигнала

Тот же сигнал дифференциального манчестерского кодирования в другом масштабе для детального анализа.



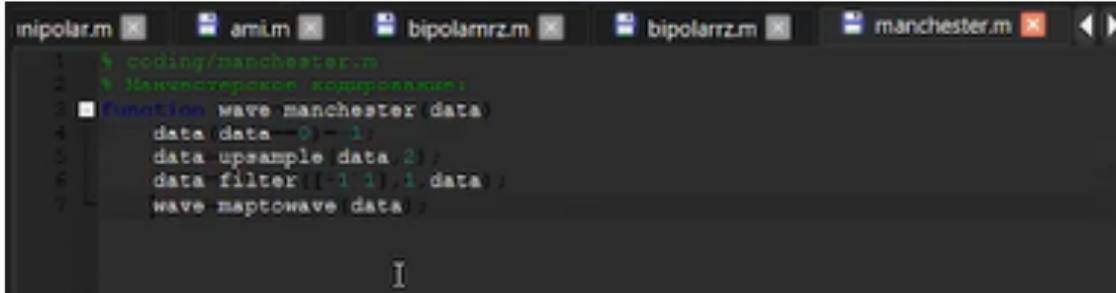
The screenshot shows a MATLAB interface with several tabs at the top: niplar.m, ami.m, bipolar.m, bipolarzz.m, manchester.m, and others. The main window displays the following MATLAB code:

```
1 % coding/manchester.m
2 % Манчестерское кодирование:
3 function wave manchester(data)
4     data = data - 1;
5     data = upsample(data, 2);
6     data = filter([-1 1], 1, data);
7     wave = maptowave(data);
```

Рис. 25: Diff. Manchester (масштаб)

График кодирования RZ

Показан трёхуровневый код с обязательным возвратом к нулю после каждого битового импульса.



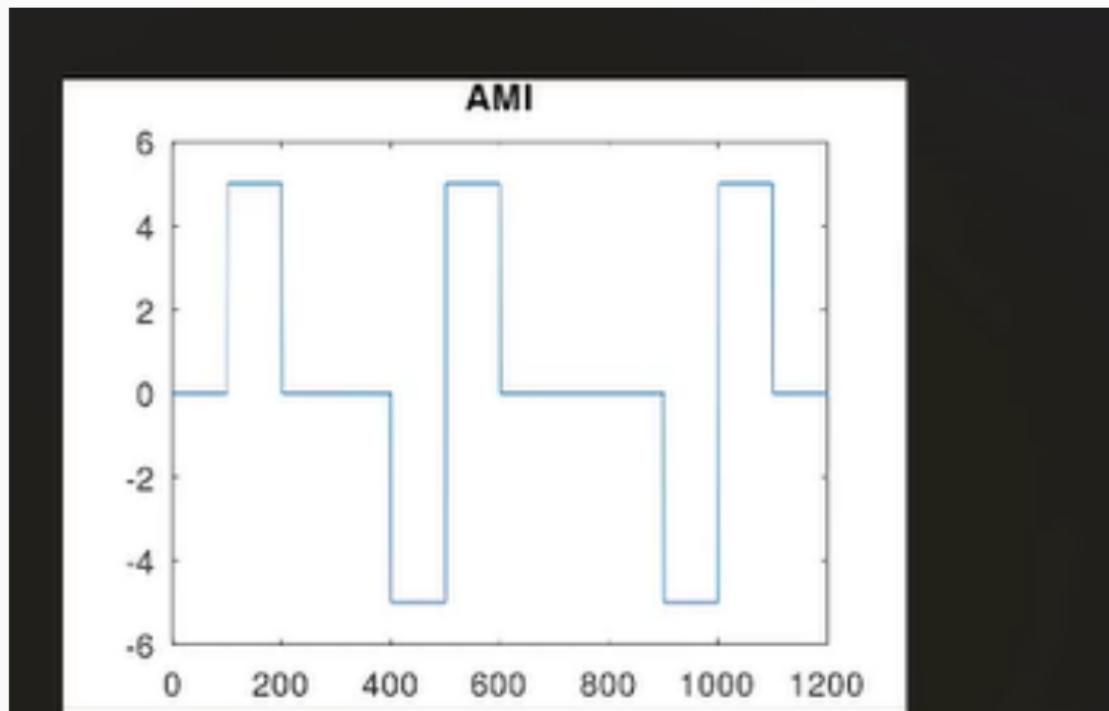
```
inipolar.m amil.m bipolamrz.m bipolarrz.m manchester.m
1 % coding/manchester.m
2 % Манчестерское кодирование:
3 function wave manchester data)
4     data = 0:-1;
5     data = upsample(data, 2);
6     data = filter([-1 1], 1, data);
7     wave = maptowave(data);
```

Рис. 26: Сигнал RZ

Часть 5: Спектральные характеристики

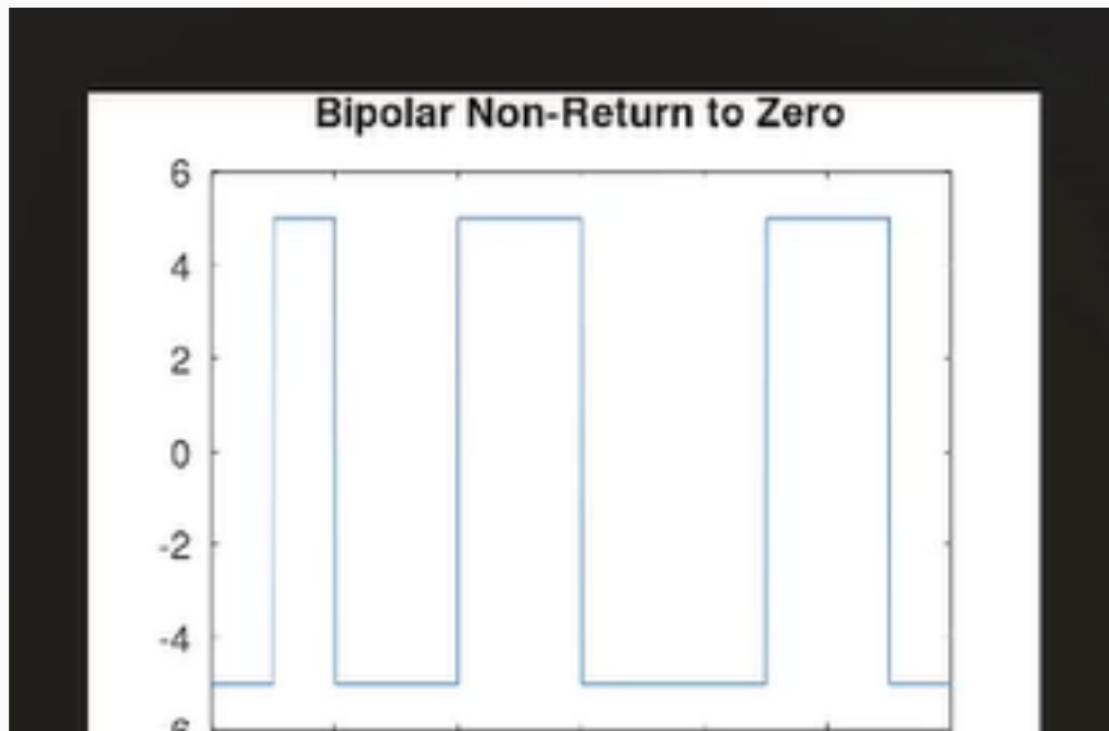
Спектр кодирования AMI

Анализ распределения энергии. Отсутствие пика на нулевой частоте (отсутствие постоянной составляющей).



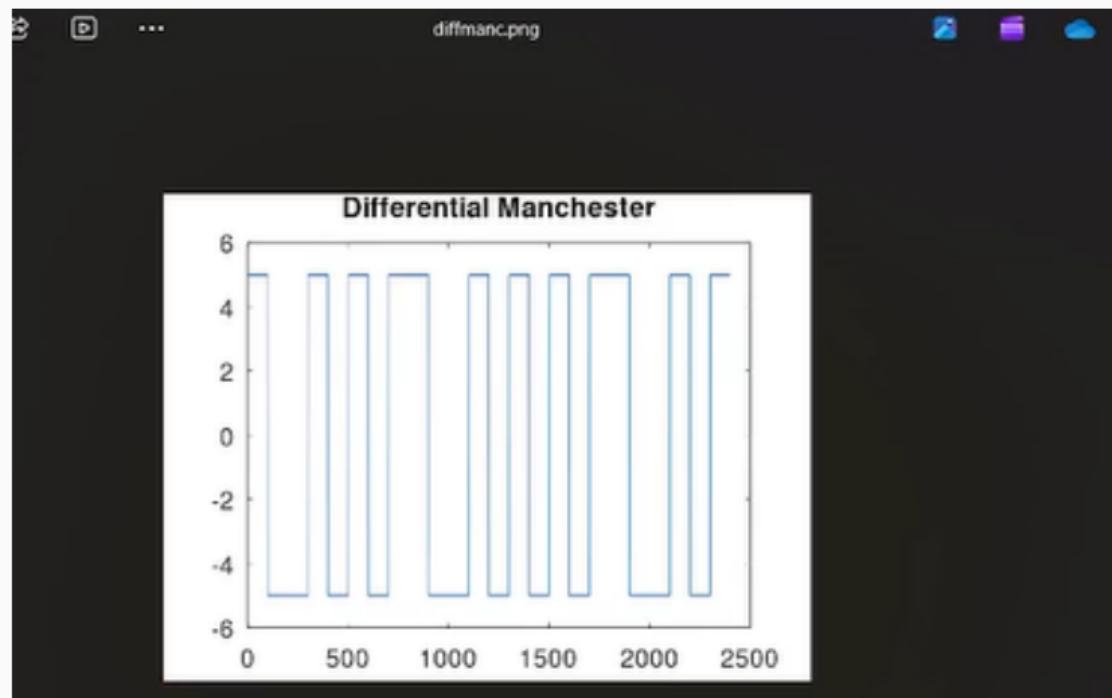
Спектр кодирования NRZ

Сравнение с другими кодами: NRZ обладает наиболее узкой полосой пропускания, но имеет постоянную составляющую.



Спектр кодирования RZ

Анализ спектра RZ-кода. Из-за возврата к нулю полоса частот становится шире, чем у NRZ.



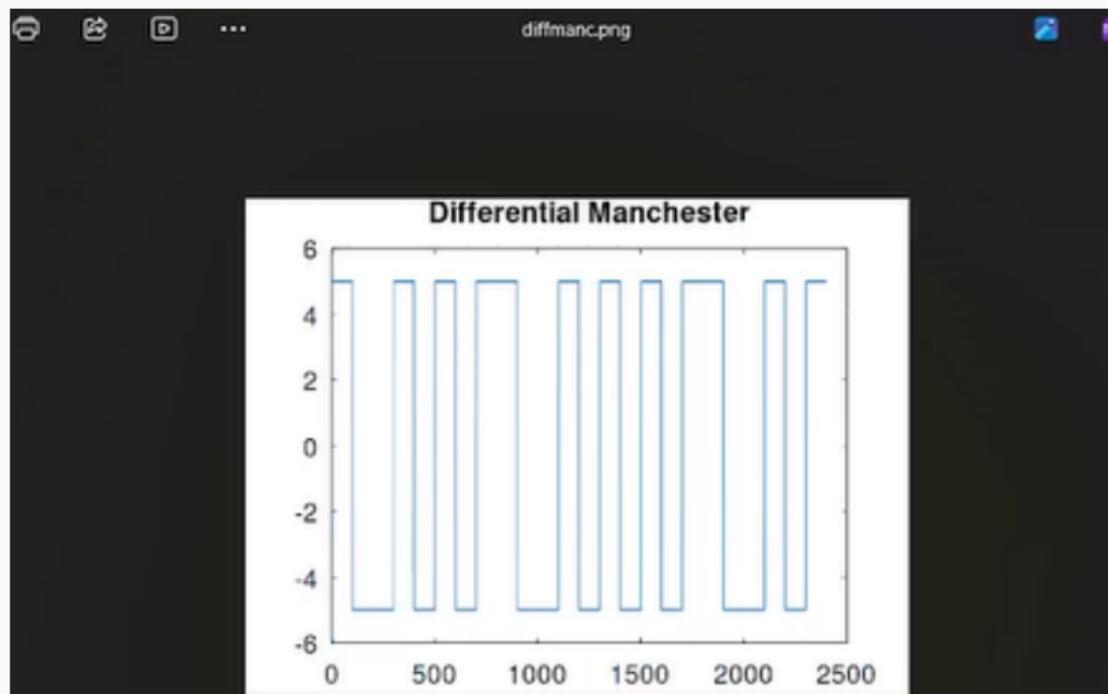
Спектр Diff. Manchester

Изучение спектральных свойств метода, широко используемого в сетевых технологиях (например, Token Ring).



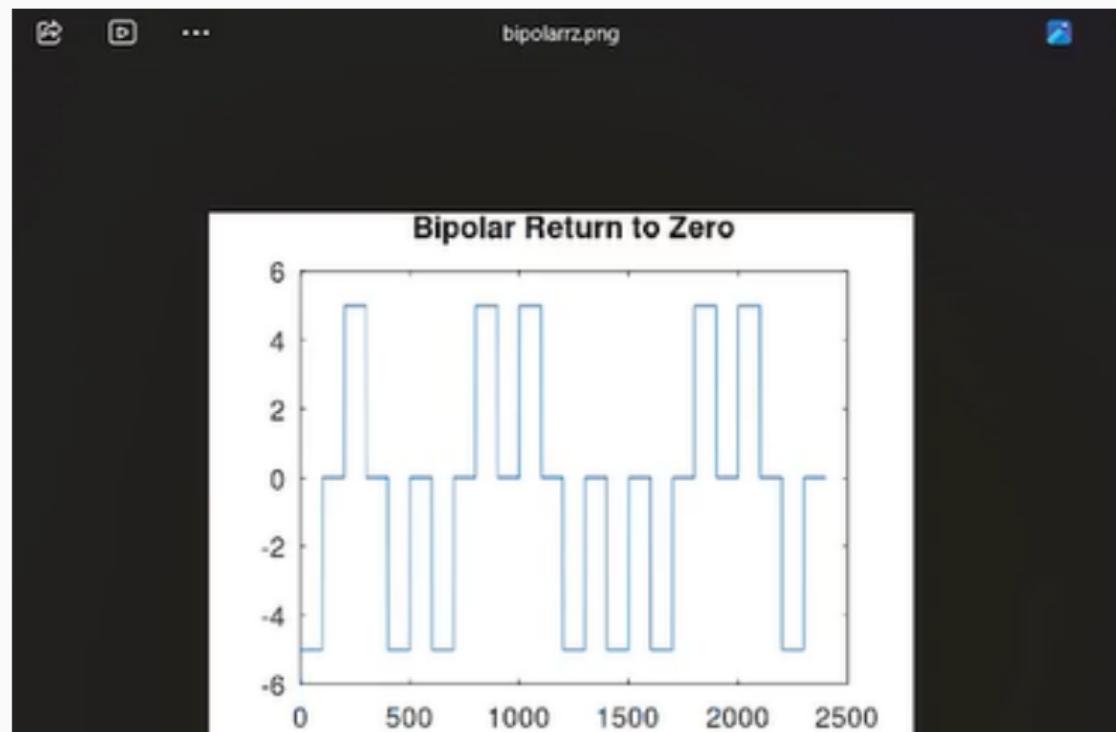
Спектр манчестерского кода

Показывает значительную энергию на высоких частотах, что важно учитывать при выборе физической среды.



Спектр униполярного кода

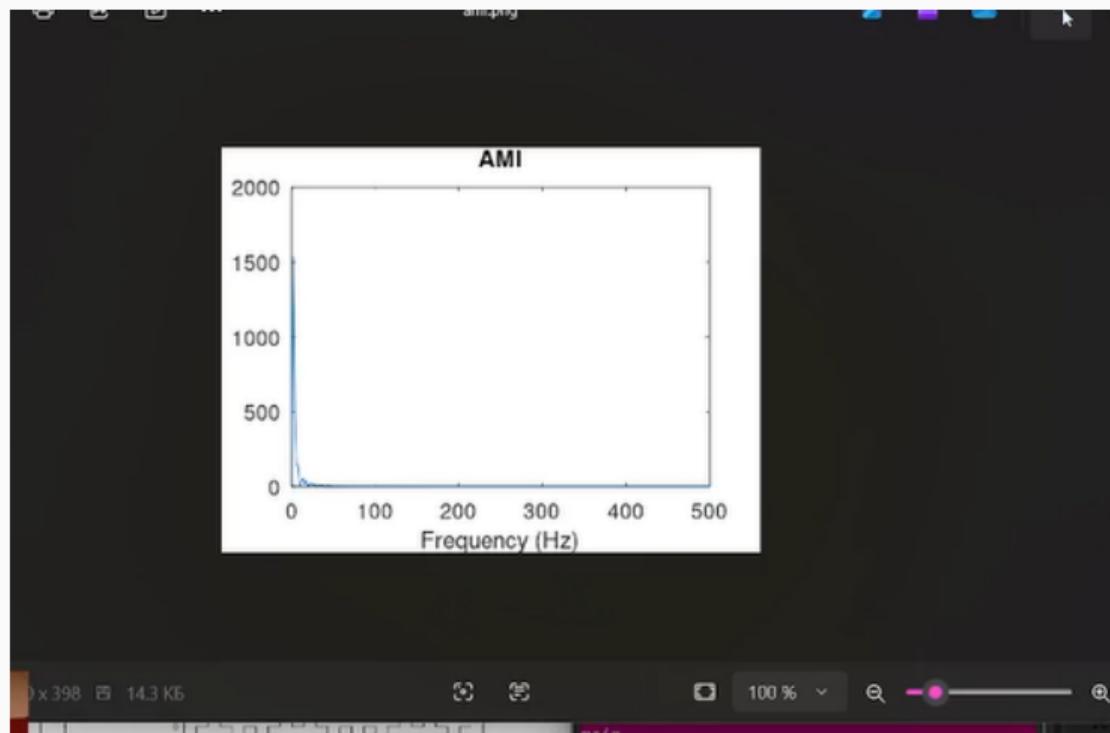
Базовый пример спектра простейшего кода для сравнения эффективности современных методов.



Часть 6: Свойства самосинхронизации

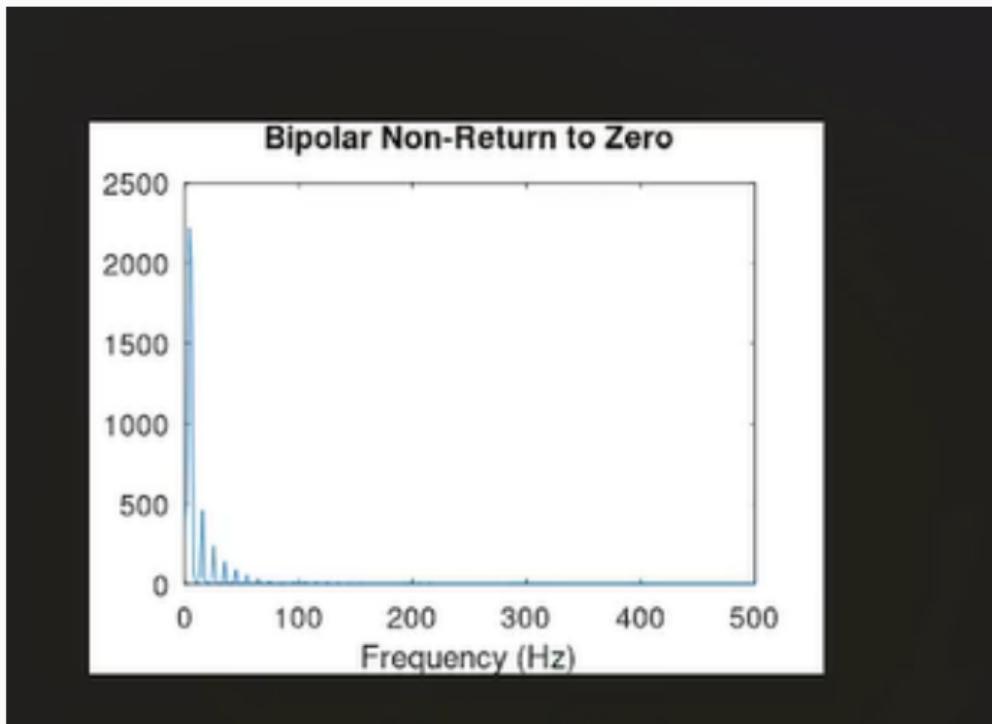
Самосинхронизация AMI

Исследование: AMI сохраняет синхронизацию только при наличии единиц.
При длинной серии нулей сигнал затухает.



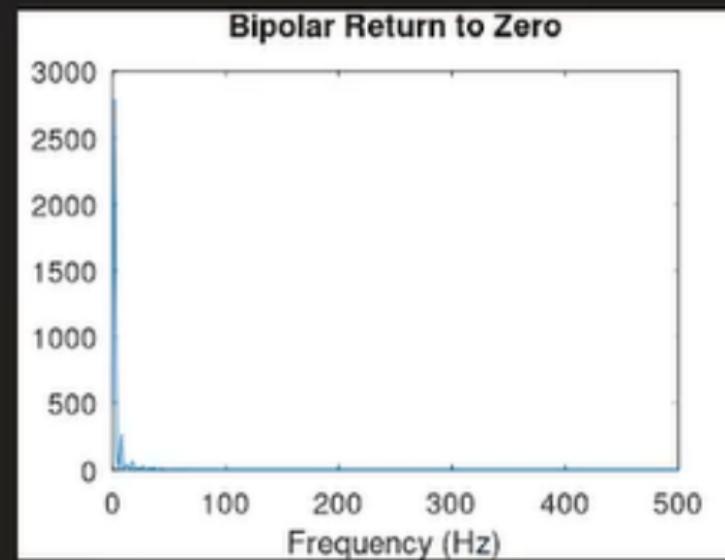
Самосинхронизация NRZ

Демонстрация отсутствия самосинхронизации: при длинной серии одинаковых бит сигнал не меняется.



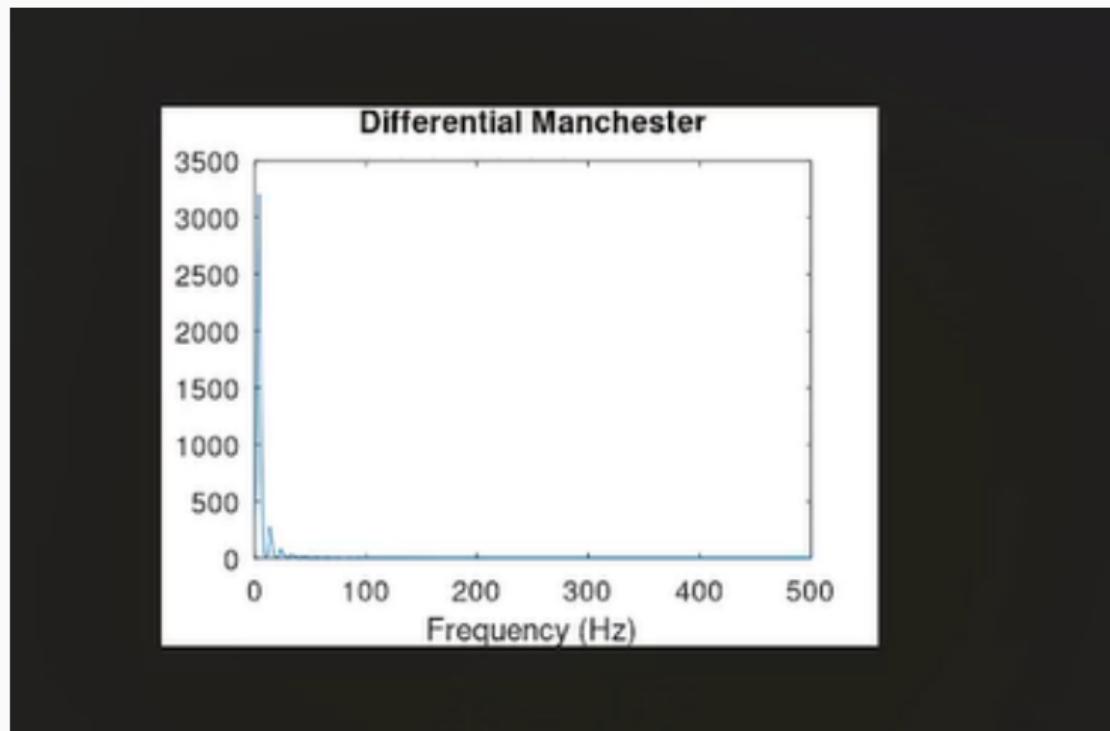
Самосинхронизация RZ

Благодаря возврату к нулю в каждом такте для единиц, код RZ обеспечивает лучшую синхронизацию.



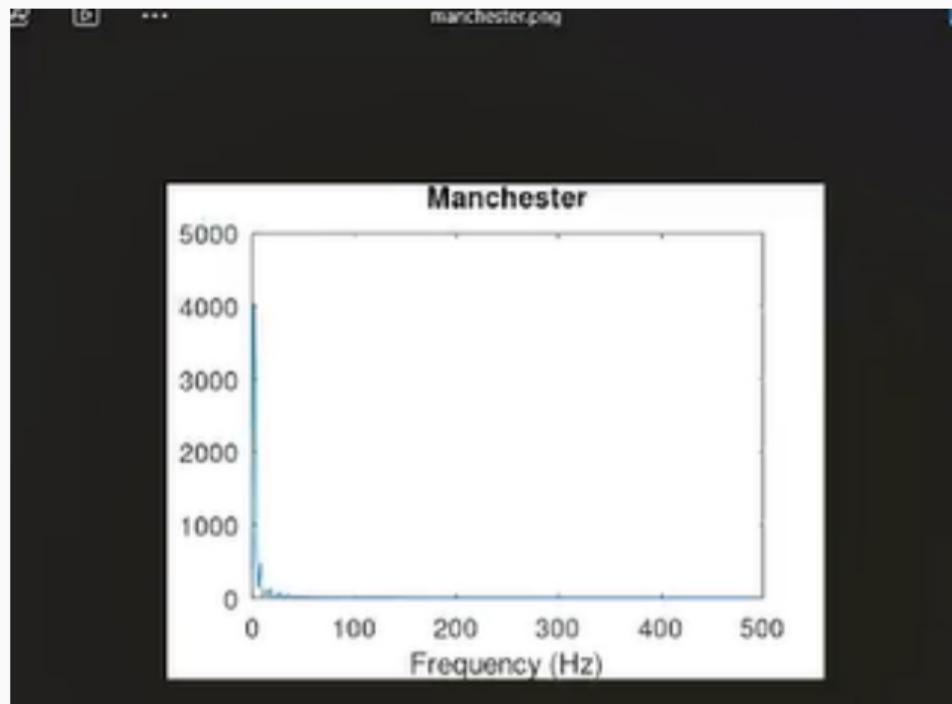
Синхронизация Diff. Manchester

Подтверждение того, что метод обеспечивает самосинхронизацию независимо от передаваемых данных.



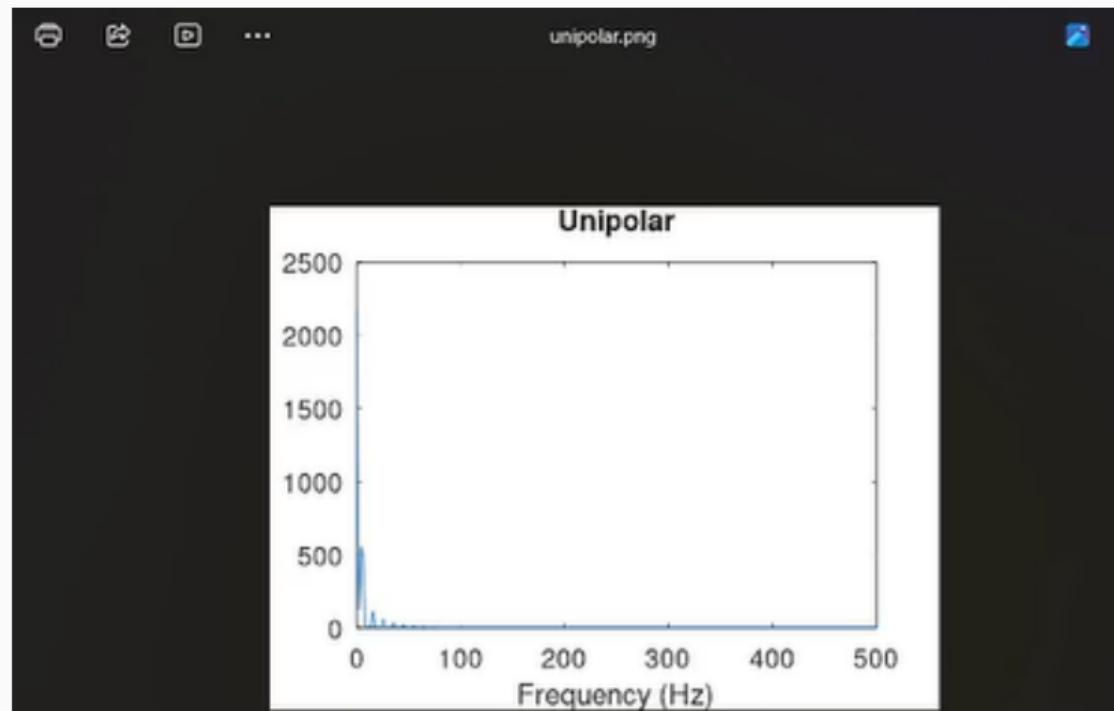
Синхронизация Manchester

Обязательные переходы в центре каждого бита гарантируют надежное выделение тактовой частоты.



Синхронизация Unipolar

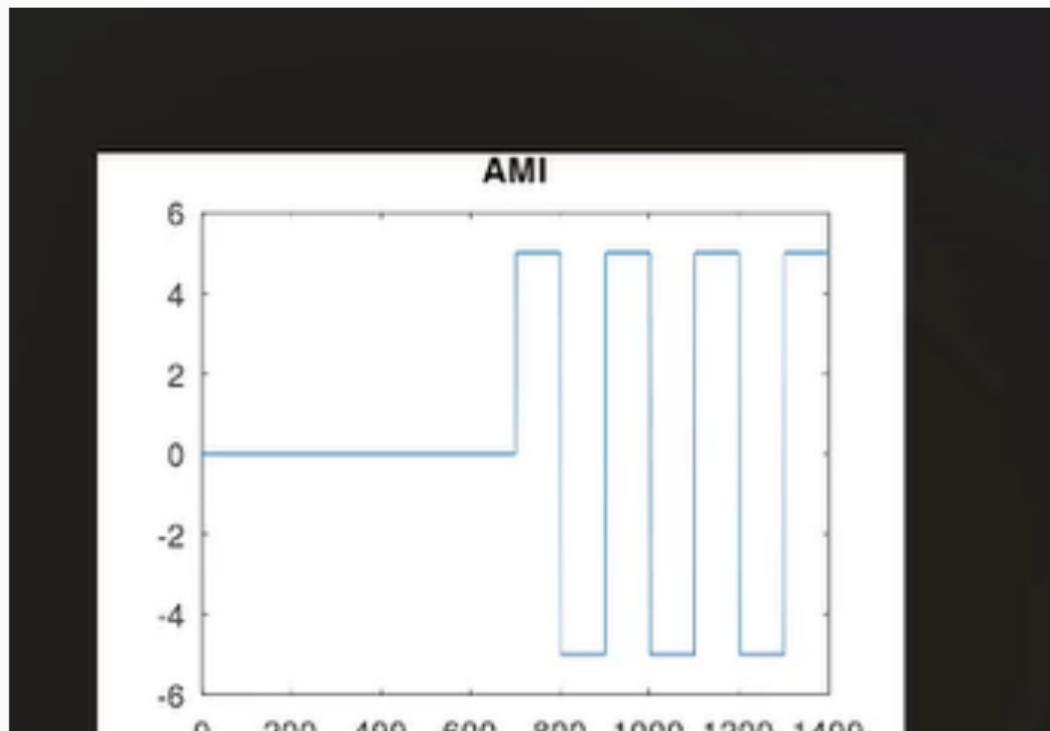
Иллюстрация полной непригодности униполярного кода для синхронизации при отсутствии переходов.



Часть 7: Дополнительные результаты

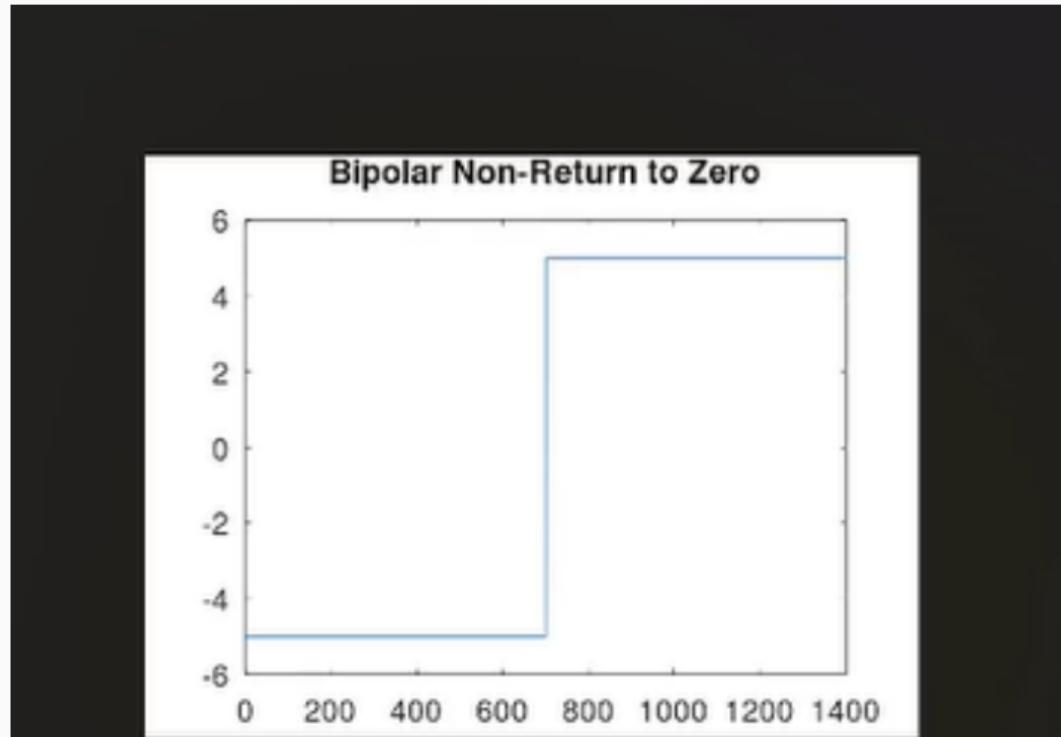
Дополнительный спектр №1

Повторный анализ спектральных характеристик для верификации полученных данных.



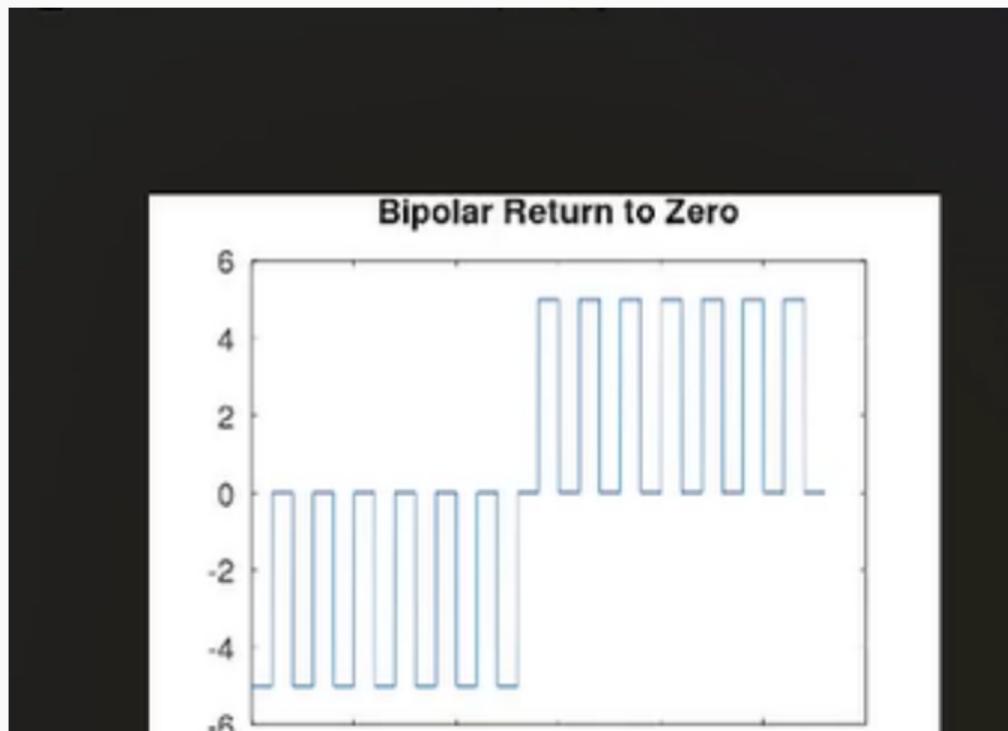
Дополнительный спектр №2

Иллюстрация результатов под другим углом для уточнения формы боковых лепестков.



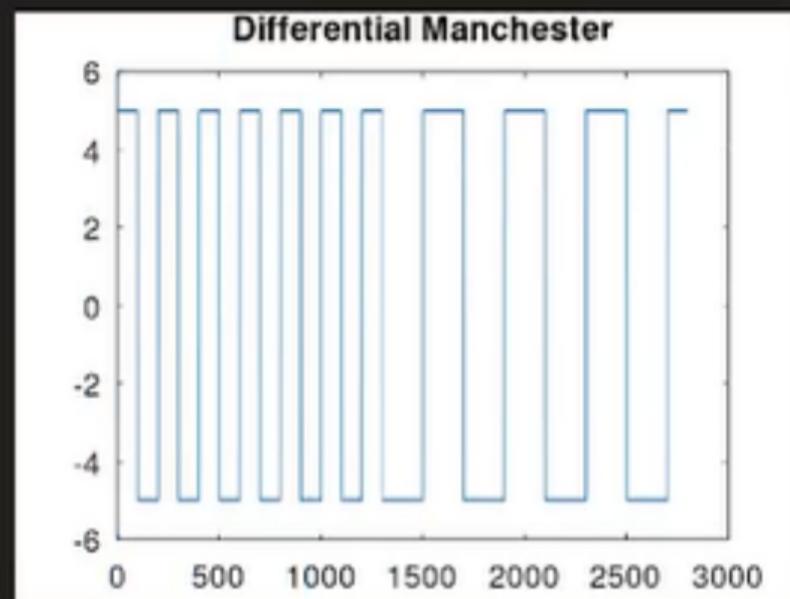
Дополнительный спектр №3

Сравнение энергетических характеристик сигналов в различных форматах экспорта.



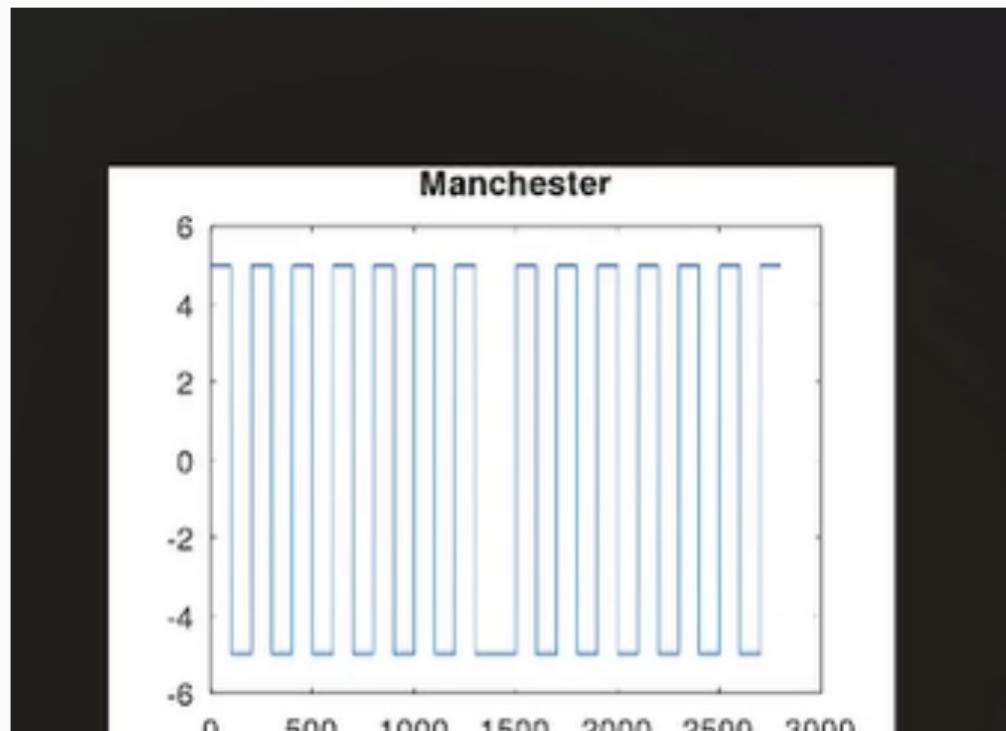
Уточнение временной диаграммы №1

Дополнительный график кодирования для анализа переходных процессов.



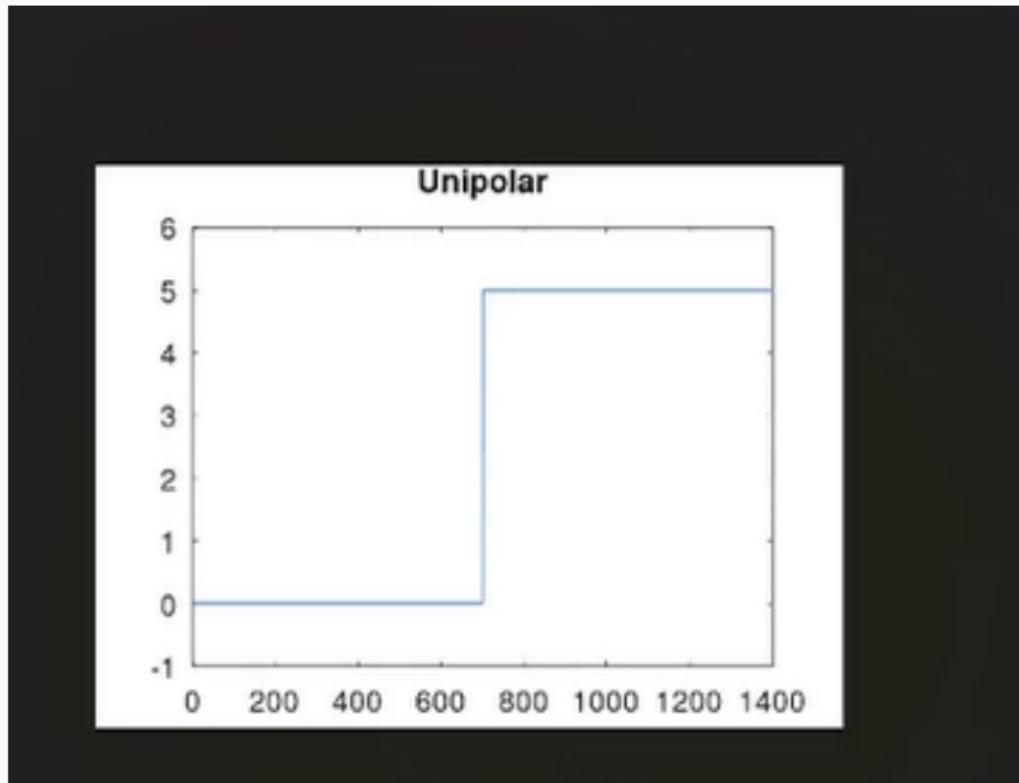
Уточнение временной диаграммы №2

Временная развертка сигнала для подтверждения корректности работы m-файлов.



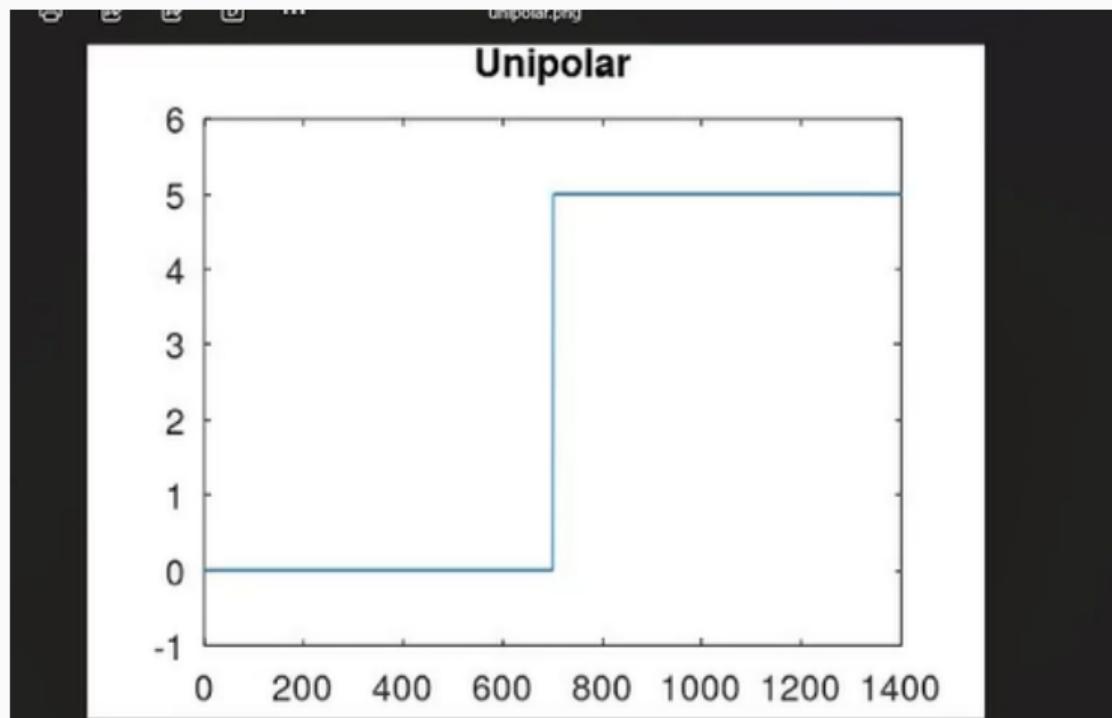
Уточнение временной диаграммы №3

Финальная проверка формирования кодированных последовательностей.



Сводный график результатов

Итоговая визуализация всех проведенных экспериментов в рамках лабораторной работы.



Заключение

Выводы по работе

- Изучены методы разложения сигналов в ряд Фурье и БПФ в среде Octave.
- Реализована модель амплитудной модуляции и проанализирован её спектр.
- На практике доказано, что манчестерские коды обеспечивают лучшую самосинхронизацию.
- Освоены навыки спектрального анализа цифровых и аналоговых сигналов.