Сетевые технологи

Лабораторная работа № 1. Методы кодирования и модуляция сигналов

Демидова Екатерина Алексеевна

Содержание

# 1 Постановка задачи

Изучить методы кодирования и модуляции сигналов с помощью высокоуровнего языка программирования Octave. Определить спектр и параметры сигнала. Продемонстрировать принципы модуляции сигнала на примере аналоговой амплитудной модуляции. Исследовать свойства самосинхронизации сигнала

# 2 Выполнение лабораторной работы

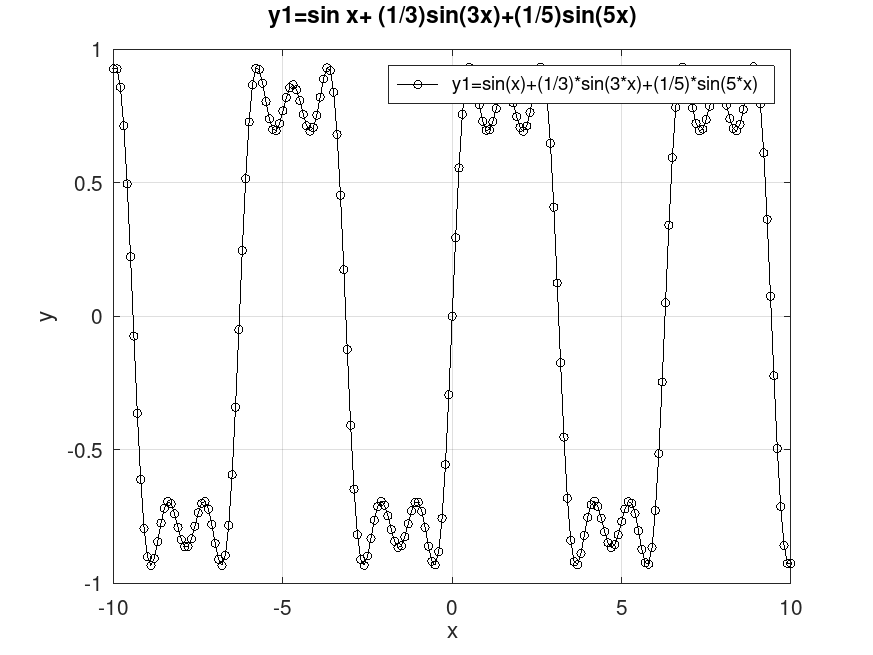
## 2.1 Построение графиков в Octave

Построим график функции в Octave. Для этого повторим данный в описании лабораторной работы листинг. В нём сначала формируются массивы и и задаётся функция, график которой мы хотим получить. Строим график с помощью функции plot, задав в параметрах, что график будет сплошной линией с маркерами размера 4 формы незакрашенный круг (листинг 1).

**Листинг 1. График**

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

В результате получим следующий график (рис. ??)



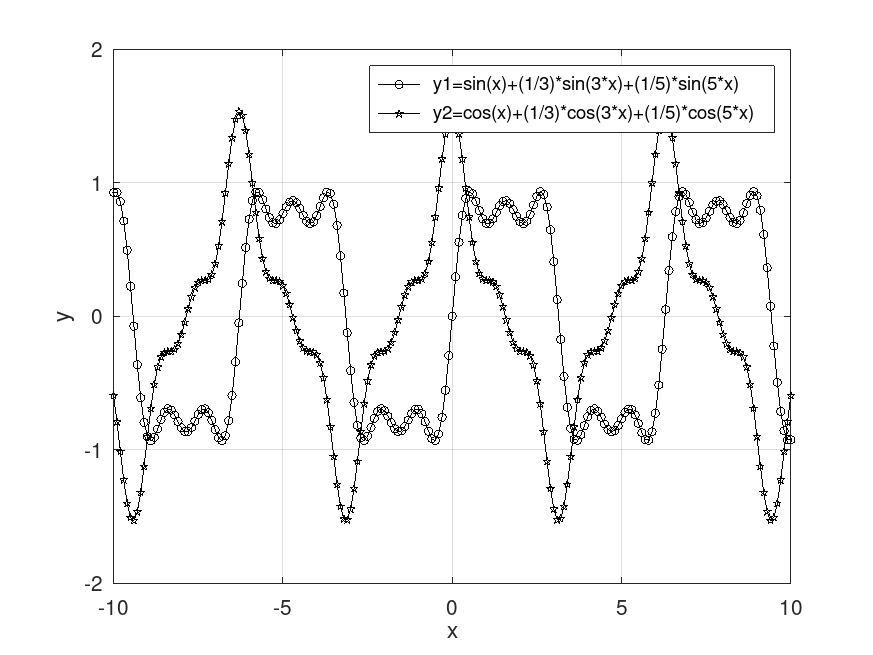
График

Теперь построим график предыдущей функции совместно с графиком функции . Для этого в Листинге 1 после команды построения первого графика добавим команду hold on и команду построения второго графика, задав в качестве маркера незакрашенный квадрат (листинг 2).

**Листинг 2. Графики и**

% 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 первой функции:  
plot(x,y1, "-ok; y1=sin(x)+(1/3)\*sin(3\*x)+(1/5)\*sin(5\*x);","markersize",4)  
% Заблокируем режим очистки окна  
hold on;  
% Построение графика второй функции:  
plot(x,y2, "-pk; 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');  
% Экспорт рисунка в файл .eps:  
print ("plot-sin2.eps", "-mono", "-FArial:16", "-deps")  
% Экспорт рисунка в файл .png:  
print("plot-sin2.png");

В результате получим следующий график (рис. ??).



Графики и

## 2.2 Разложение импульсного сигнала в частичный ряд Фурье

Разработаем код m-файла, результатом выполнения которого являются графики меандра, реализованные с различным количеством гармоник.

Для этого используем листинг, данный в лабораторной работе. В нём сначала задаются количество гармоник, частота дискретизации, амплитуда и период. Затем задаётся вектор-строка, состоящая из нечётных номеров гармоник в спектре. Амплитуды гармоник, образующих меандр, обратно пропорционалы номеру соответствующей гармоники в спектре. Также для построения графиков через формулу с косинусами реализуем чередование знака амплитуд. Теперь создадим массив гармоник, а с помощью умножения его на вектор, состоящий из амплитуд гармоник, получим массив элементов ряда. 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. В конце экспортируем полученный график с помощью команды plot (листинг 3).

**Листинг 3. Построение графиков меандра, содержащего различное число гармоник, через косинус**

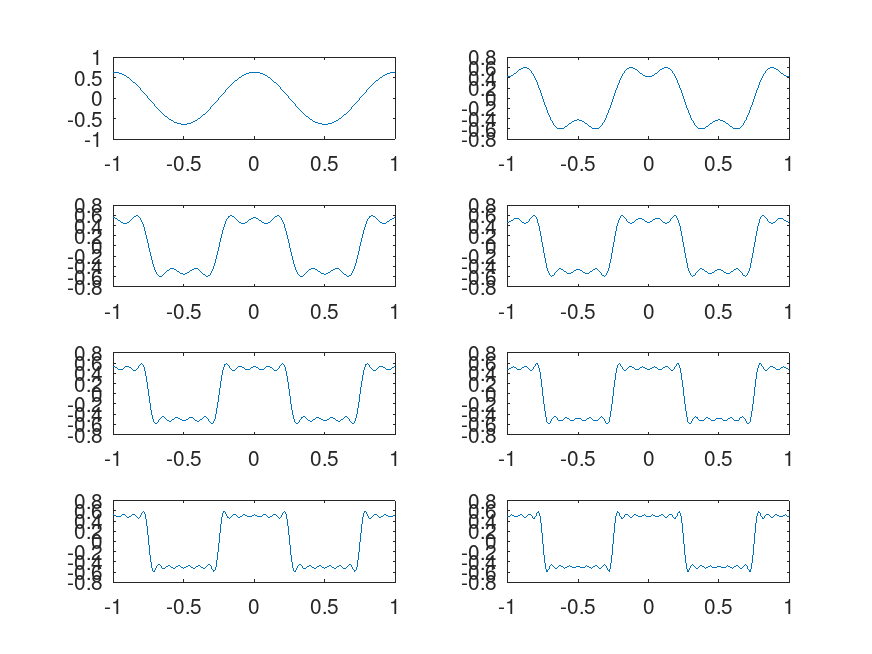
% количество отсчетов (гармоник):  
N=8;  
% частота дискретизации:  
t=-1:0.01:1;  
% значение амплитуды:  
A=1;  
% период:  
T=1;  
% амплитуда гармоник  
nh=(1:N)\*2-1;  
% массив коэффициентов для ряда, заданного через cos:  
Am=2/pi ./ nh;  
Am(2:2:end) = -Am(2:2:end);  
% массив гармоник:  
harmonics=cos(2 \* 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  
print("meandr\_cos.png");

Затем реализуем построение графиков меандр через синус, для этого из листинга 3 уберём строчку, в которой реализуется чередование знака в формуле и заменим функцию косинуса на функцию синуса при создании массива гармоник (листинг 4).

**Листинг 4. Построение графиков меандра, содержащего различное число гармоник, через синус**

% 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;  
% 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  
print("meandr\_sin.png")

В результате обеих программ получим следующий график (рис. ??).



Графики меандра, содержащего различное число гармоник

## 2.3 Определение спектра и параметров сигнала

Определим спектр двух отдельных сигналов и их суммы. Также выполним это задание с другой частотой дискретизации и определим, что будет, если взять частоту дискретизации меньше 80 Гц.

Зададим начальные значения: длину сигнала, частоту дискретизации, частоты первого и второго сигналов и их амплитуды, а также массив отсчётов времени и спектр сигнала. Затем зададим два синусоидальных сигнала разной частоты и построим графики сигналов (листинг 5).

**Листинг 5. Построение графиков двух сигналов**

% 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';

В результате получим следующий график (рис. ??).

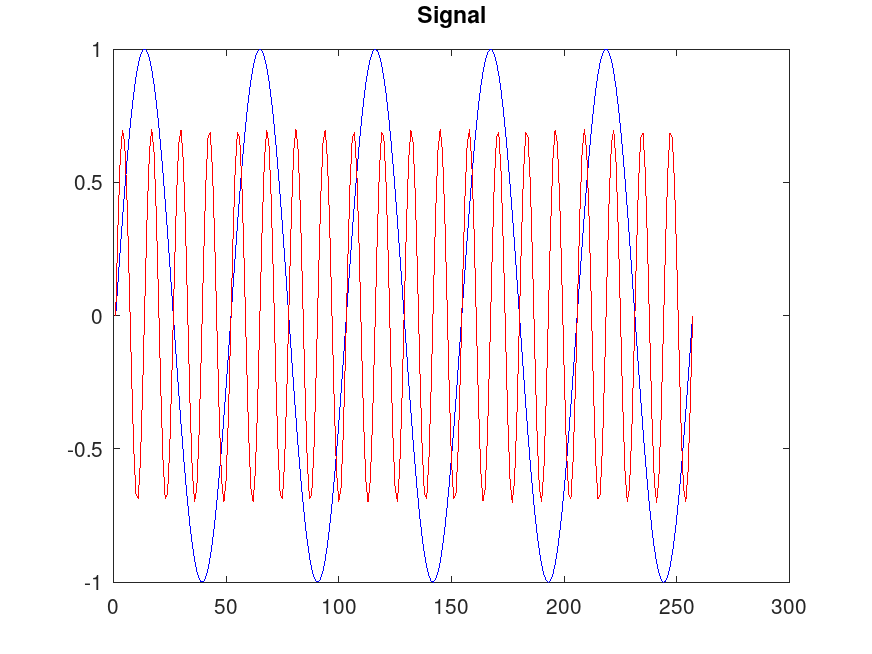


График двух синусоидальных сигналов разной частоты

С помощью быстрого преобразования Фурье(которое реализовано функцией fft()) найдём спектры сигналов, добавив в листинг 4 следующий код(листинг 5):

**Листинг 5. Построение графика спектров синусоидальных 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';

В результате получим следующий график (рис. ??).

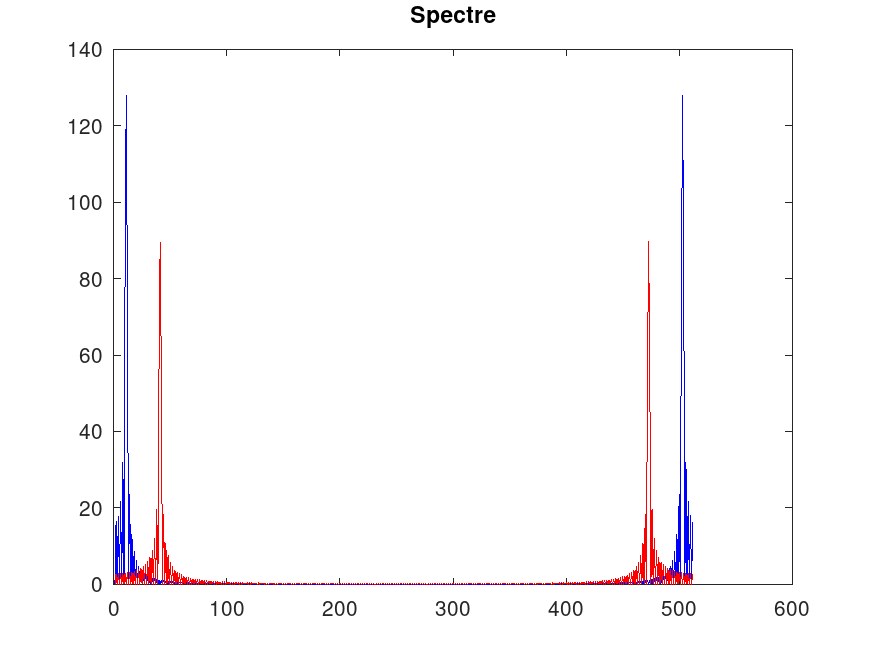


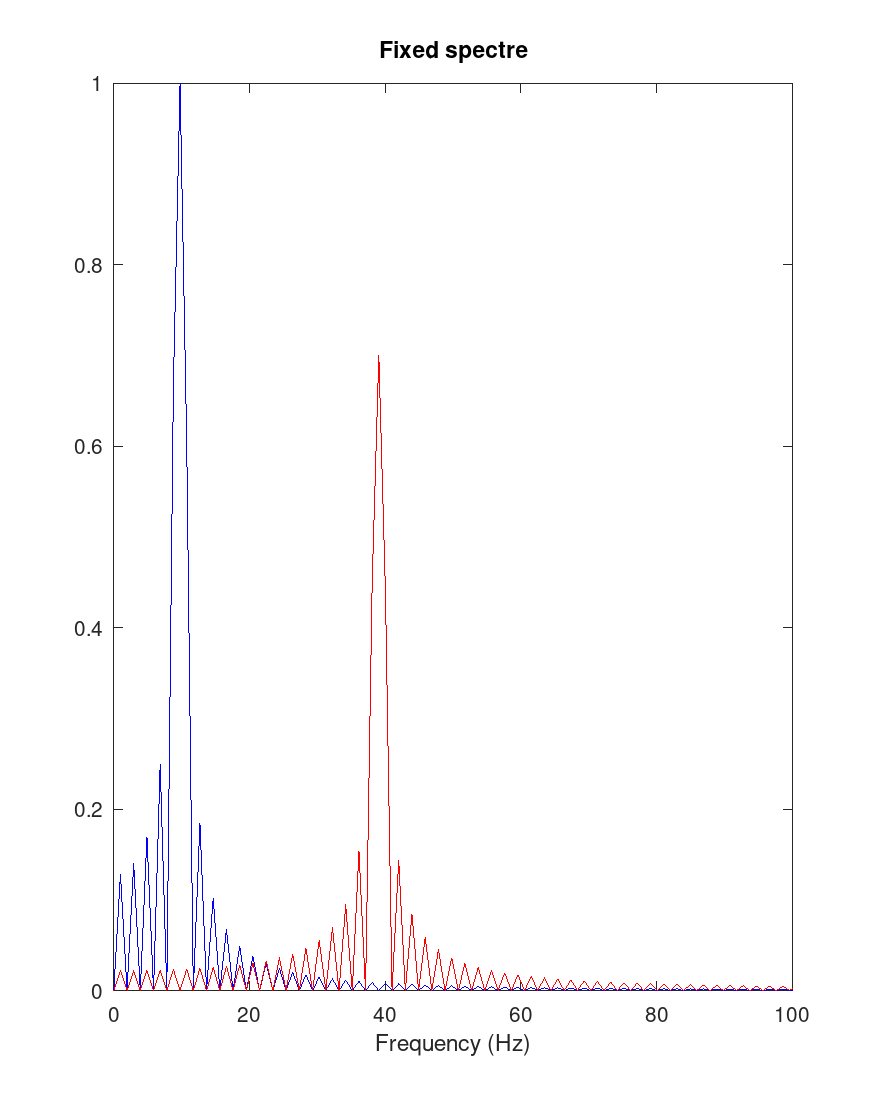
График спектров синусоидальных сигналов

Скорректируем график спектра (рис. 5): отбросим дублирующие отрицательные частоты, а также примием в расчёт то, что на каждом шаге вычисления быстрого преобразования Фурье происходит суммирование амплитуд сигналов. Для этого зададим сетку частот(частоты будут принимать значения от 0 до 250) и проведём нормировку спектров по амплитуде(значения по оси ординат будут от 0 до 1), добавив к предыдущему коду следующую часть(листинг 6):

**Листинг 6. 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';

В результате получим следующий график (рис. ??).



Откорректированнвй график спектров синусоидальных сигналов

Найдём спектр суммы рассмотренных сигналов(листинг 7).

**Листинг 7. Спектр суммы сигналов**

% 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 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 графика спектра сигнала:  
plot(f,spectre(1:fd2+1))  
xlim([0 100]);  
title('Spectre');  
xlabel('Frequency (Hz)');  
print 'spectre/spectre\_sum.png';

В результате получим график суммарного сигнала (рис. ??):

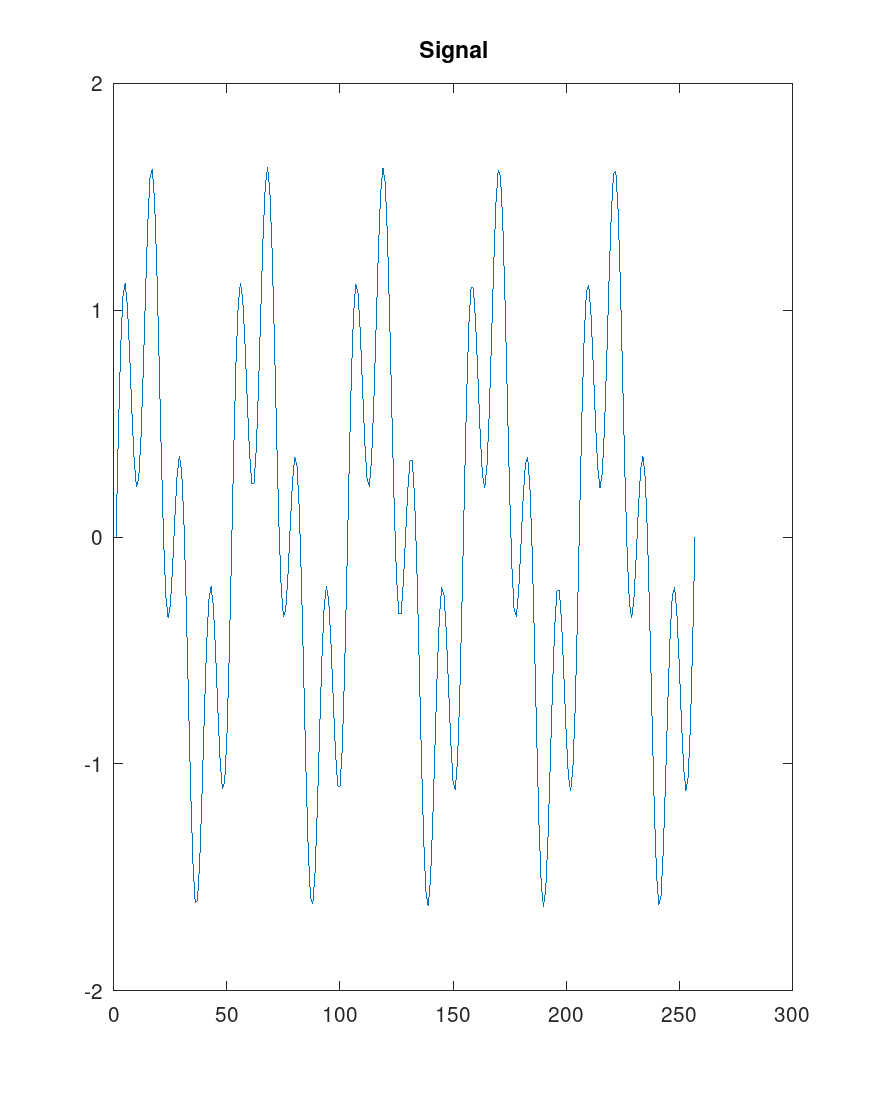


График суммарного сигнала

А также так как спектр суммы сигналов должен быть равен сумме спектров сигналов, что вытекает из свойств преобразования Фурье, получим аналогичный предыдущему результат(рис. ??):

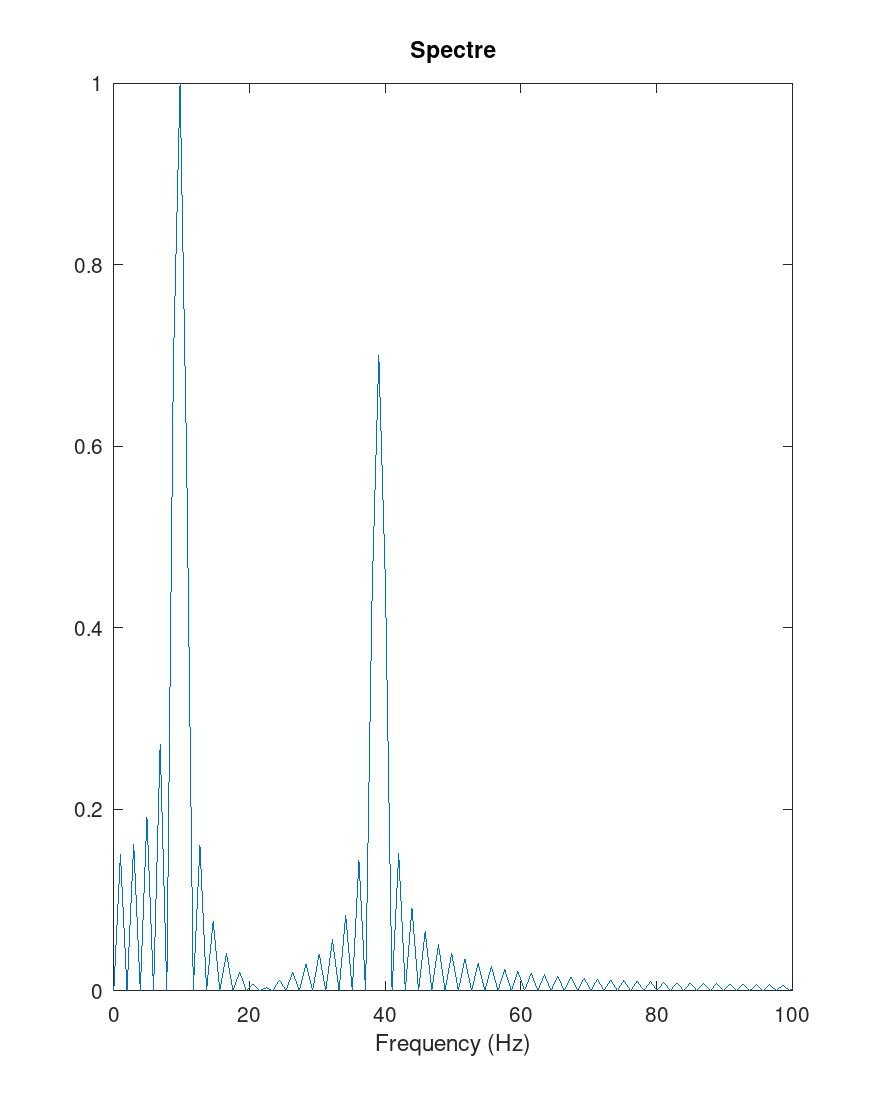


График спектра суммарного сигнала

Выполним это задание с другой частотой дискретизации, Для этого в листингах 5 и 7 заменим значение переменной fd на 128. В результате получим следующие графики(рис. ??-??):

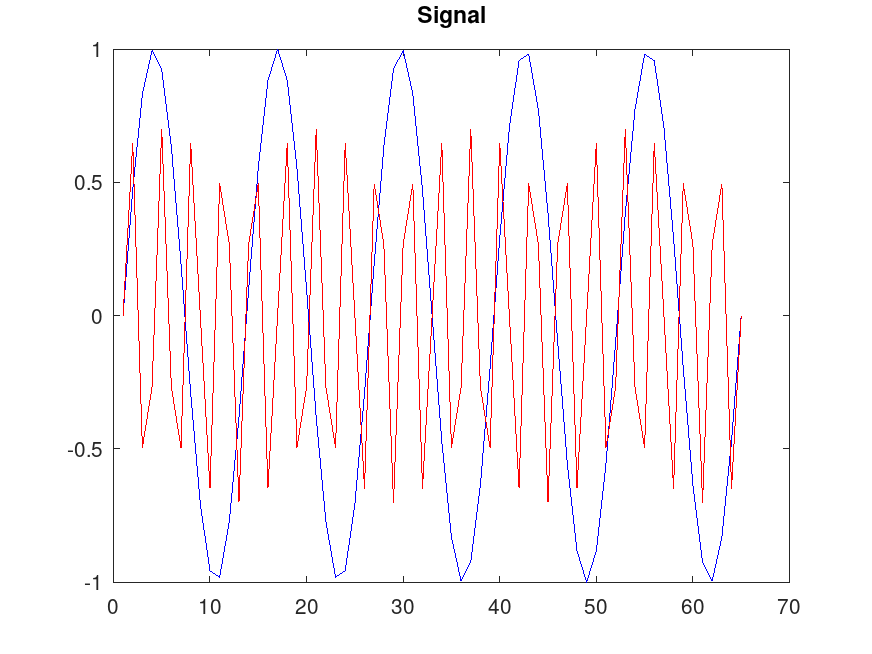
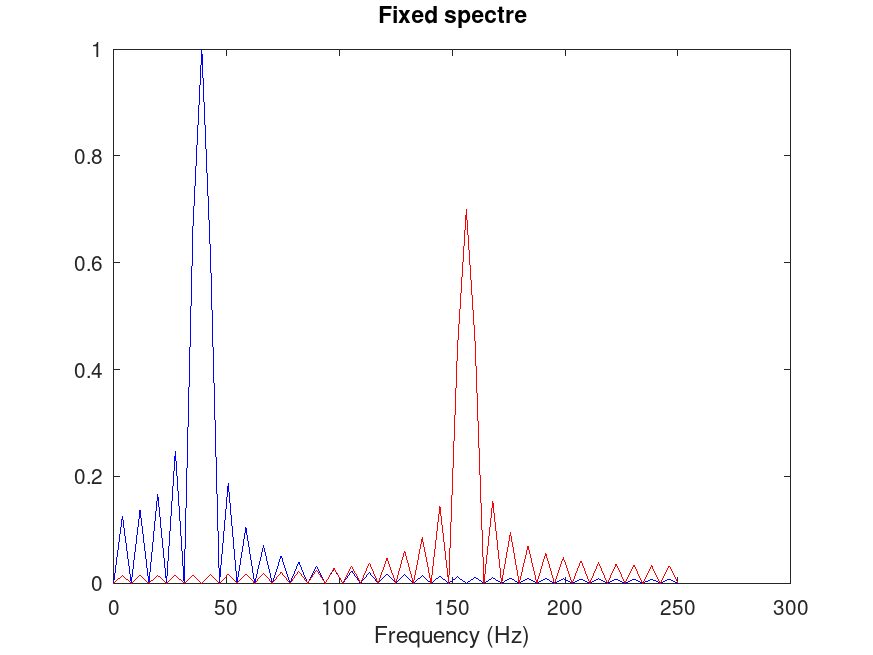


График двух синусоидальных сигналов разной частоты при fd = 128.



Откорректированнвй график спектров синусоидальных сигналов при fd = 128.

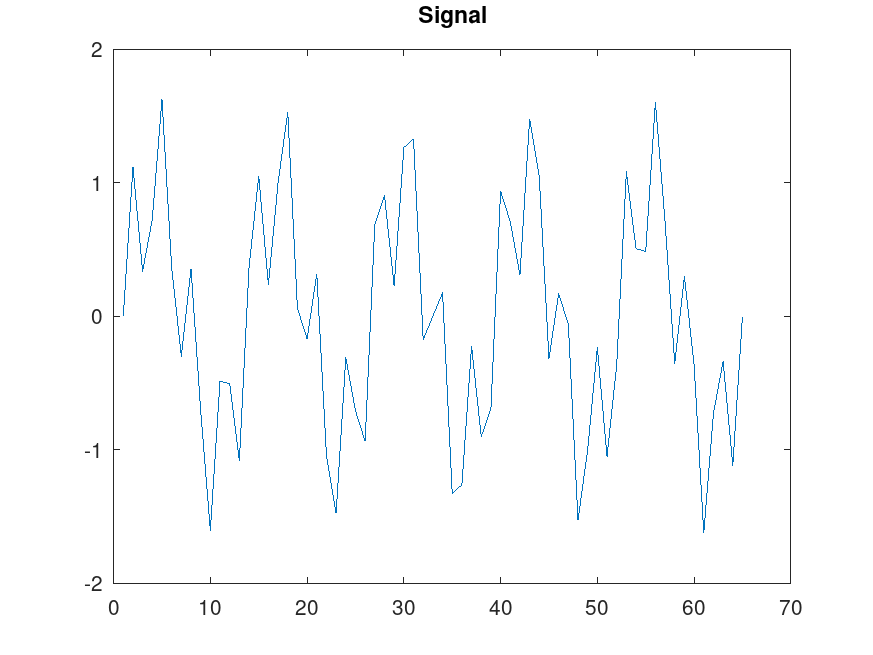


График суммарного сигнала при fd = 128.

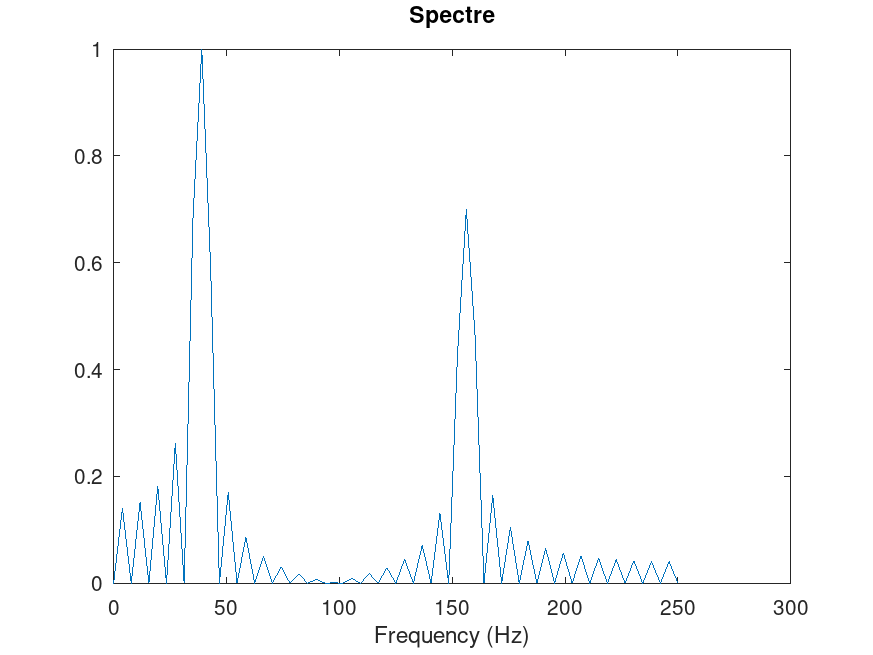


График спектра суммарного сигнала при fd = 128.

При частоте дисктреризации максимумальные амплитуды спектров сигналов находились на значениях частот приблизительно равных частотам самих сигналов и графики самих сигналов имели вид синусов. При уменьшении частоты дискретизации точки соответствующие максимальным амплитудам на графике спектров сдвинулись в право по оси частот, а графики сигналов не соответствуют графикам синусоидальной функции. То есть чем меньше частота дискретизации, тем менее точно можно найти графики сигналов, их спектров, суммарного сигнала и спектра суммарного сигнала.

Проверим, что будет, если взять частоту дискретизации меньше 80 Гц. Для этого в листингах 5 и 7 заменим значение переменной fd на 70. В результате получим следующие графики(рис. ??-??):

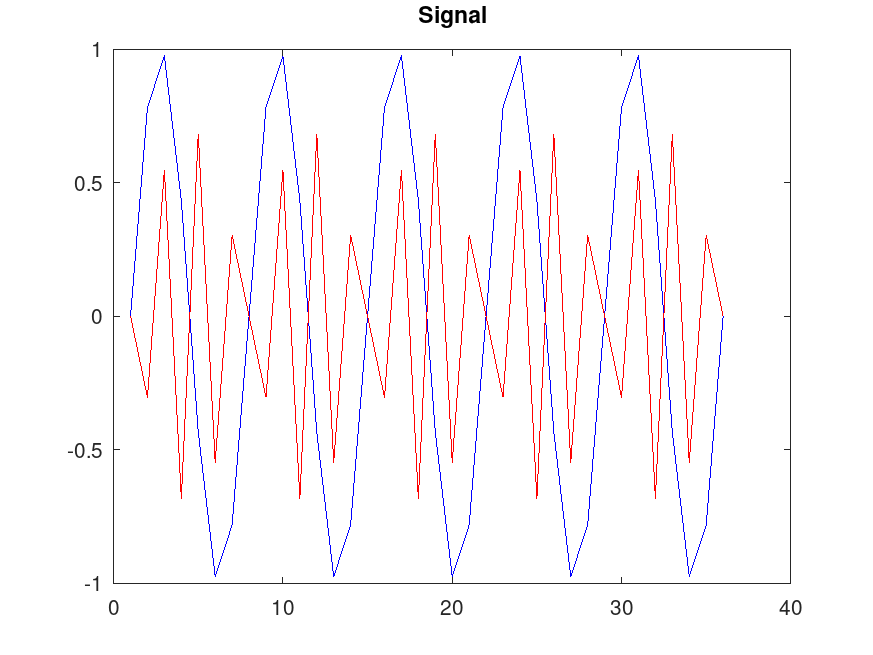
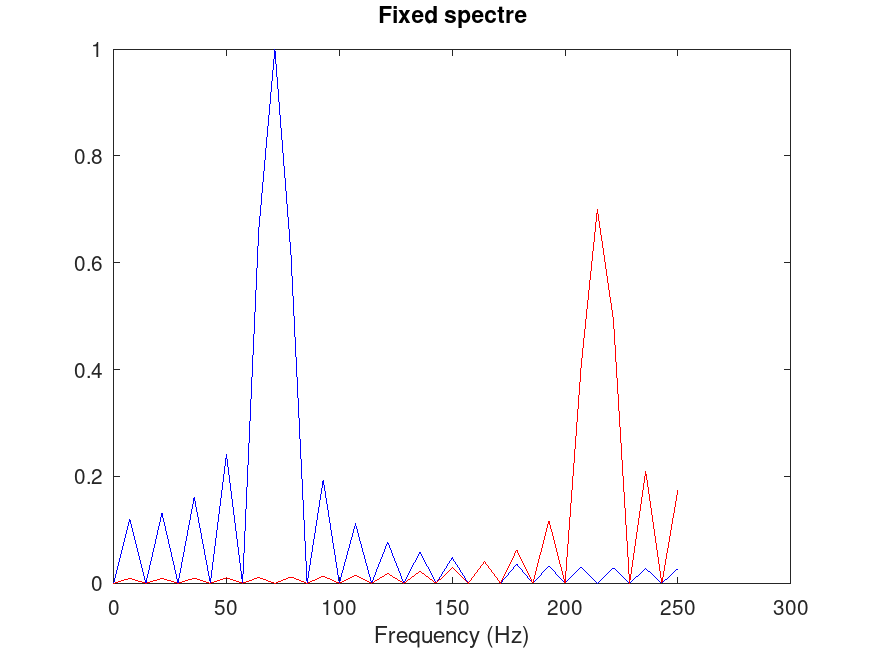


График двух синусоидальных сигналов разной частоты при fd = 70.



Откорректированнвй график спектров синусоидальных сигналов при fd = 70.

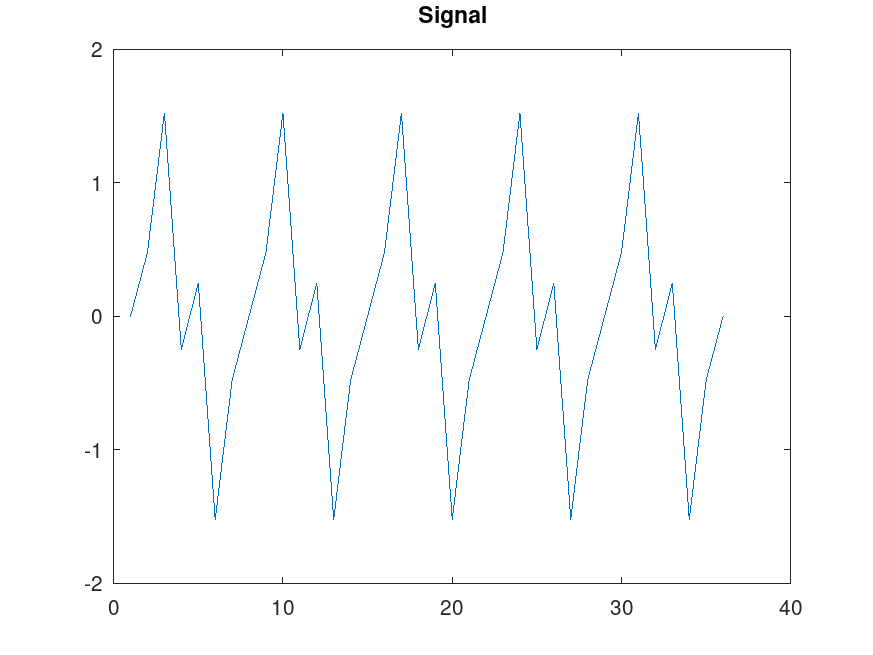


График суммарного сигнала при fd = 70.

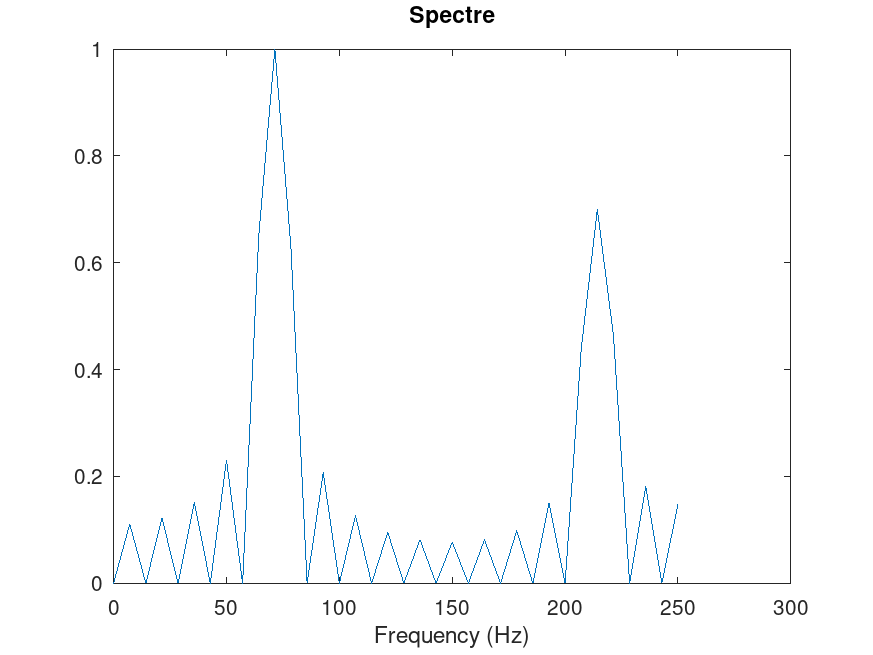


График спектра суммарного сигнала при fd = 70.

При значении частоты дискретизации меньше 80 Гц сигнал, обозначенный красным цветом на графике, не соответсвует действительности. Он имеет частоту 40 Гц. По теореме Котельникова сигнал может быть восстановлен однозначно и без потерь по своим гармоникам, взятым с частотой, строго большей удвоенной верхней частоты. Соответственно, если взять частоту дискретизации не больше 10, то оба сигнала не смогут быть восстановлены по своим спектрам.

## 2.4 Амплитудная модуляция

Покажем принципы модуляции сигнала на примере аналоговой амплитудной модуляции(листинг 8).

**Листинг 8. Демонстрация аналоговой амплитудной модуляции**

% 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:  
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';

В результате получим график сигнала и огиббающей при амплитудной модуляции(рис. ??):

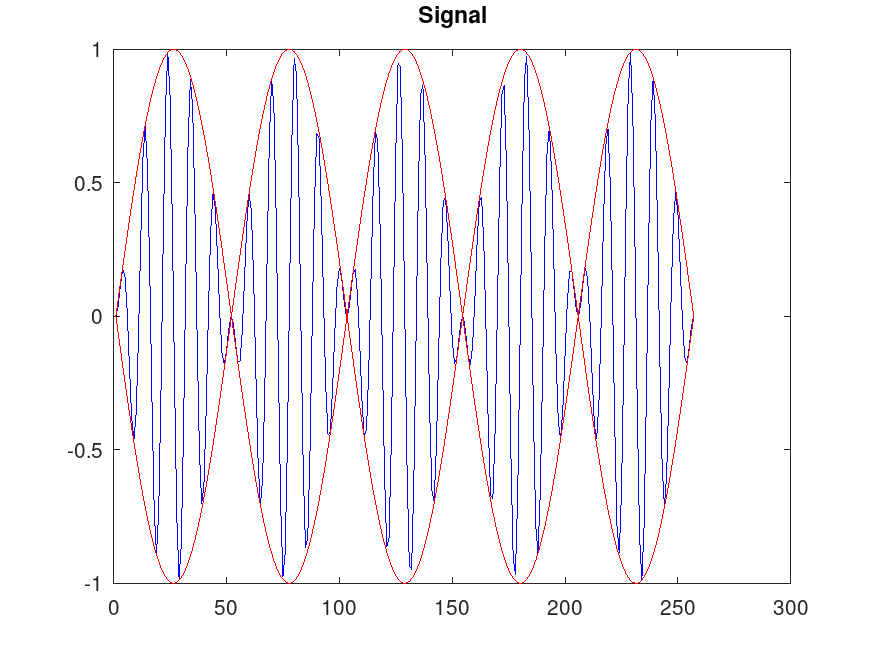


График сигнала и огибающей при амплитудной модуляции

В результате получаем, что спектр произведения представляет собой свёртку спектров (рис. ??):

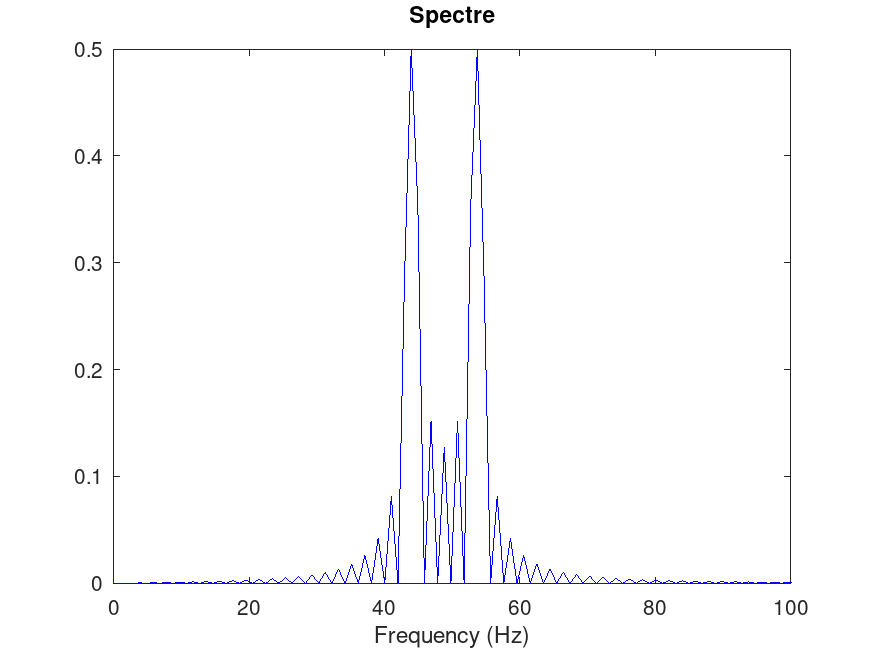


График спектра сигнала при амплитудной модуляции

## 2.5 Кодирование сигнала. Исследование свойства самосинхронизации сигнала

Для заданных битовых последовательностей получим кодированные сигналы для нескольких кодов, проверим свойства самосинхронизуемости кодов, и получим спектры.

Для этого в папке coding создадим файлы main.m, maptowave.m, unipolar.m, ami.m, bipolarnrz.m, bipolarrz.m, manchester.m, diffmanc.m, calcspectre.m. Также установим пакет расширений signal.

В файле maim.m подключим пакет signal и зададим выходные кодовые последовательности (листинг 9):

**Листинг 9. Входные кодовые последовательности**

% 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 самосинхронизации:  
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");

Затем в этом же файле пропишем вызовы функций для построения графиков модуляций кодированных сигналов для кодовой последовательности data(листинг 10):

**Листинг 10. Вызовы функций для построения графиков модуляций кодированных сигналов для кодовой последовательности data**

% 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';

Затем в этом же файле пропишем вызовы функций для построения графиков модуляций кодированных сигналов для кодовой последовательности data\_sync(листинг 11):

**Листинг 11. Вызовы функций для построения графиков модуляций кодированных сигналов для кодовой последовательности data\_sync**

% Униполярное кодирование  
wave=unipolar(data\_sync);  
plot(wave);  
ylim([-1 6]);  
title('Unipolar');  
print 'sync/unipolar.png';  
  
% Кодирование AMI  
wave=ami(data\_sync);  
plot(wave)  
title('AMI');  
print 'sync/ami.png';  
% Кодирование NRZ  
wave=bipolarnrz(data\_sync);  
plot(wave);  
title('Bipolar Non-Return to Zero');  
print 'sync/bipolarnrz.png';  
  
% Кодирование RZ  
wave=bipolarrz(data\_sync);  
plot(wave)  
title('Bipolar Return to Zero');  
print 'sync/bipolarrz.png';  
% Манчестерское кодирование  
wave=manchester(data\_sync);  
plot(wave)  
title('Manchester');  
print 'sync/manchester.png';  
  
% Дифференциальное манчестерское кодирование  
wave=diffmanc(data\_sync);  
plot(wave)  
title('Differential Manchester');  
print 'sync/diffmanc.png';

Далее в этом же файле пропишем вызовы функций для построения графиков спектров(листинг 12):

**Листинг 12. Вызовы функцмй для построения графиков спектров**

% 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';

В файле maptowave.m пропишите функцию, которая по входному битовому потоку строит график сигнала(листинг 13):

**Листинг13. Функция, которая по входному битовому потоку строит график сигнала**

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

В файлах unipolar.m, ami.m, bipolarnrz.m, bipolarrz.m, manchester.m, diffmanc.m пропишем соответствующие функции преобразования кодовой последовательности data с вызовом функции maptowave для построения соответствующего графика(листинги 14-19).

**Листинг 14. Униполярное кодирование**

% coding/unipolar.m  
% Униполярное кодирование:  
function wave=unipolar(data)  
wave=maptowave(data);

**Листинг 15. Кодирование AMI**

% 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);

**Листинг 16. Кодирование NRZ**

% coding/bipolarnrz.m  
% Кодирование NRZ:  
function wave=bipolarnrz(data)  
data(data==0)=-1;  
wave=maptowave(data);

**Листинг 17. Кодирование RZ**

% coding/bipolarrz.m  
% Кодирование RZ:  
function wave=bipolarrz(data)  
data(data==0)=-1;  
data=upsample(data,2);  
wave=maptowave(data);

**Листинг 18. Манчестерское кодирование**

% coding/manchester.m  
% Манчестерское кодирование:  
function wave=manchester(data)  
data(data==0)=-1;  
data=upsample(data,2);  
data=filter([-1 1],1,data);  
wave=maptowave(data);

**Листинг 19. Дифференциальное манчестерское кодирование**

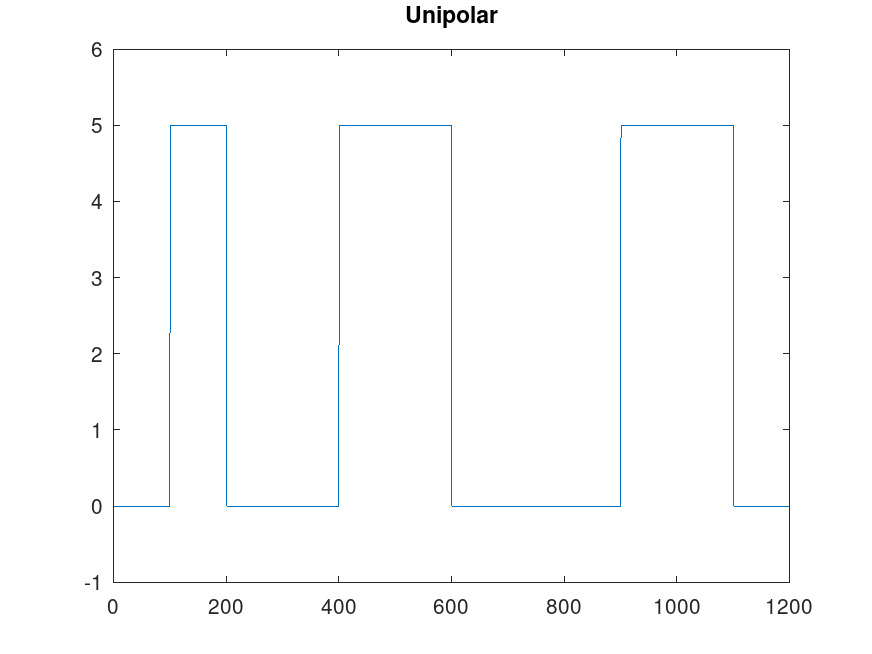
% coding/diffmanc.m  
% Дифференциальное манчестерское кодирование  
function wave=diffmanc(data)  
data=filter(1,[1 1],data);  
data=mod(data,2);  
wave=manchester(data);

В файле calcspectre.m пропишем функцию построения спектра сигнала(листинг 20):

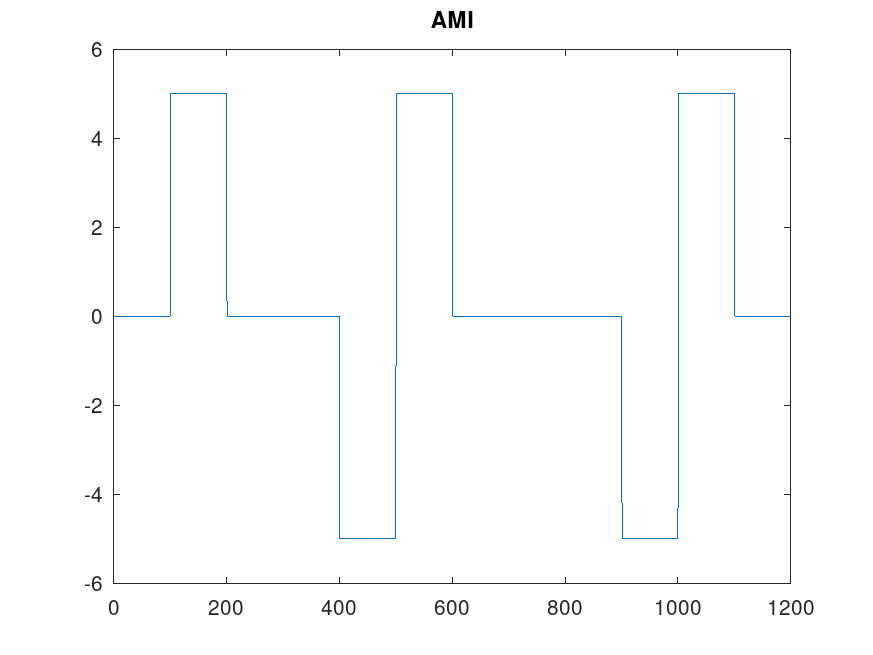
**Листинг 20. Функция построения спектра сигнала**

% calcspectre.m  
% Функция построения спектра сигнала:  
function spectre = calcspectre(wave)  
% Частота дискретизации (Гц):  
Fd = 512;  
Fd2 = Fd/2;  
Fd3 = Fd/2 + 1;  
  
X = fft(wave,Fd);  
spectre = X.\*conj(X)/Fd;  
f = 1000\*(0:Fd2)/Fd;  
plot(f,spectre(1:Fd3));  
xlabel('Frequency (Hz)');

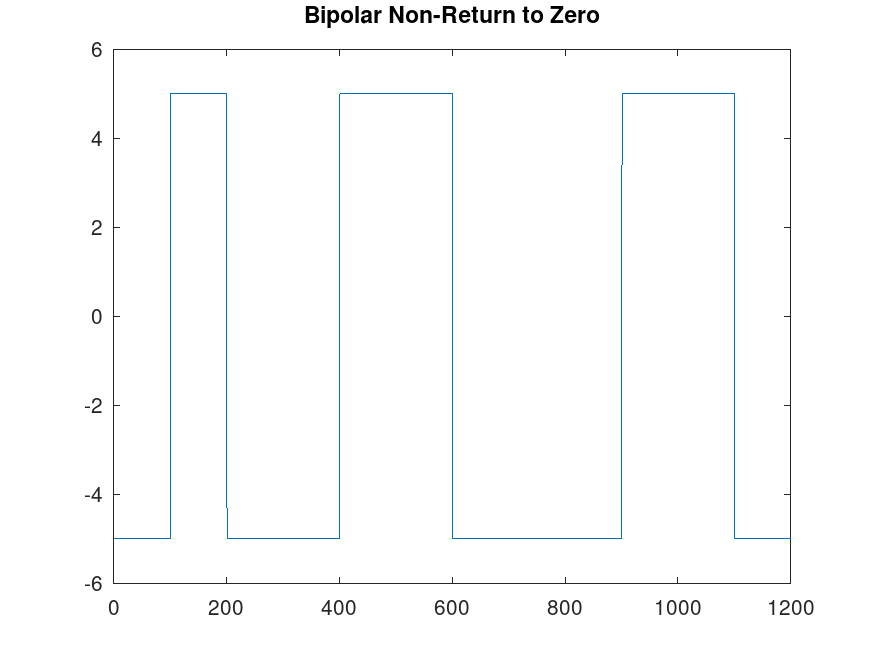
Запустим главный скрипт main.m. В каталоге signal получим файлы с графиками кодированного сигнала (рис. ??-??), в каталоге sync - файлы с графиками, иллюстрирующими свойства самосинхронизации (рис. ??-??), в каталоге spectre — файлы с графиками спектров сигналов (рис. ??-??).



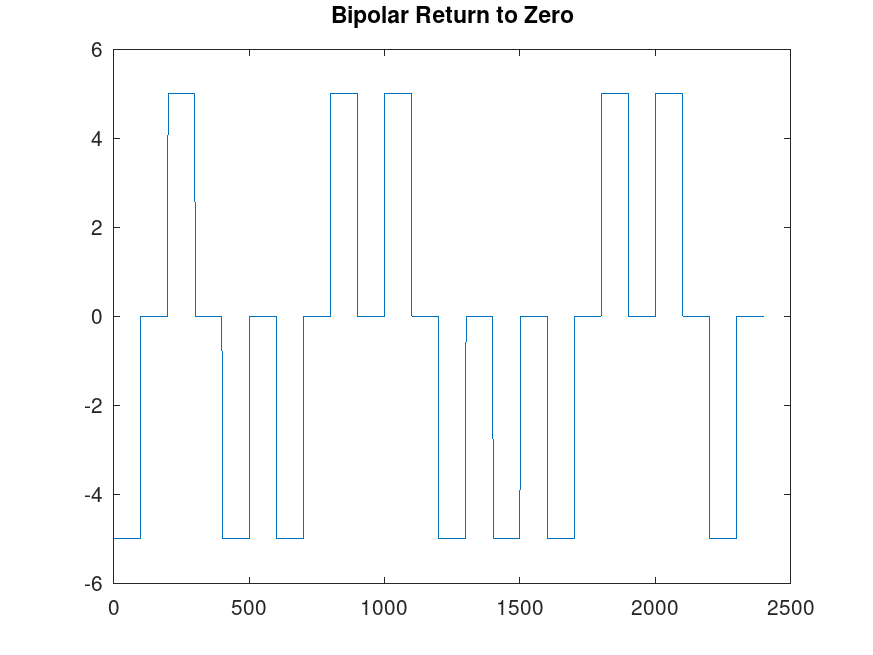
Униполярное кодирование



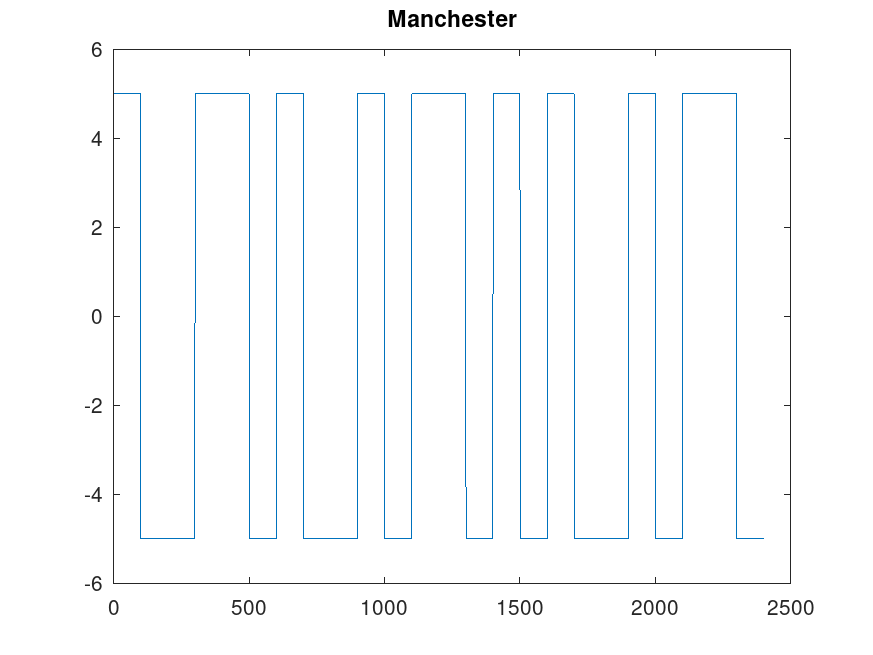
Кодирование AMI



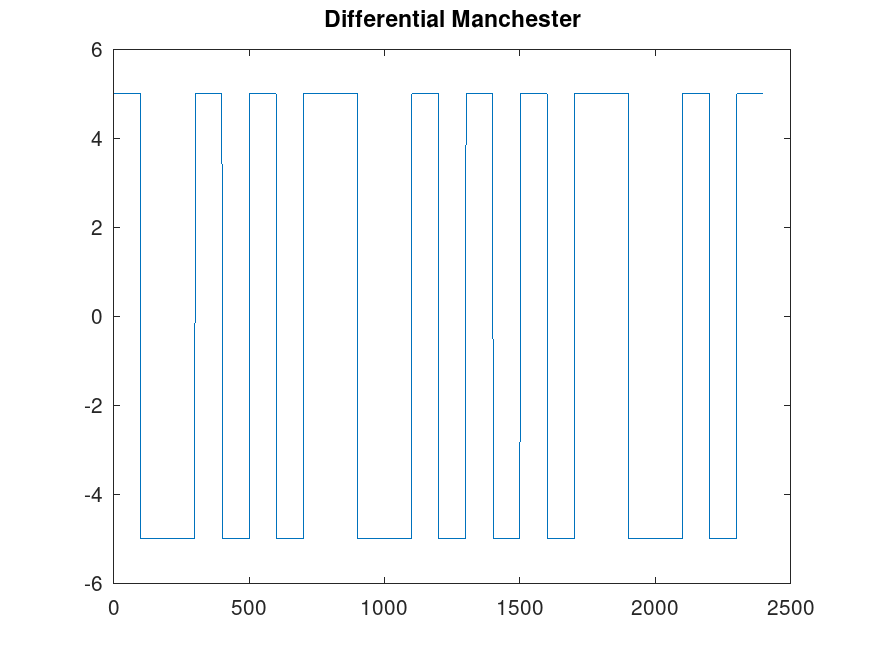
Кодирование NRZ



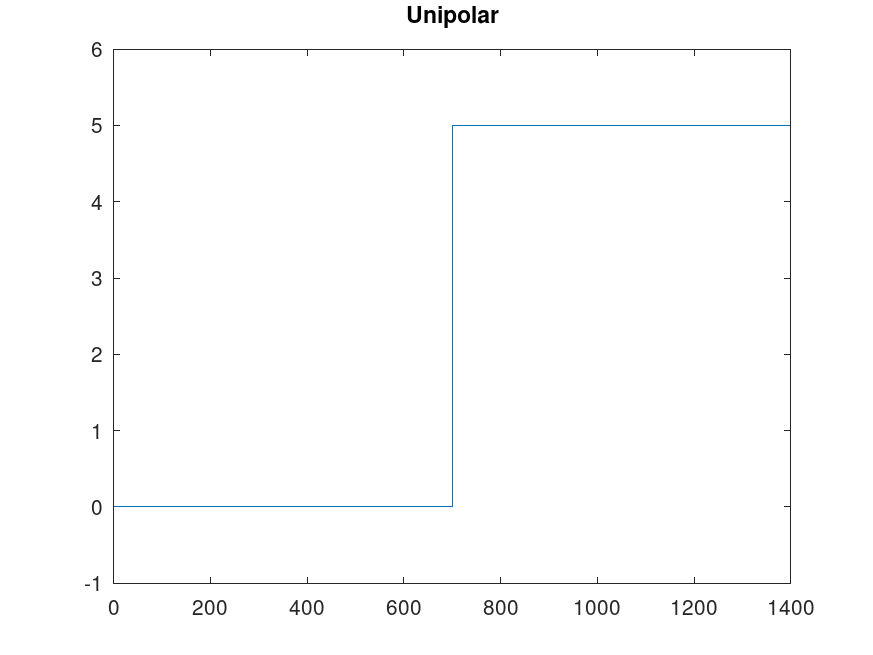
Кодирование RZ



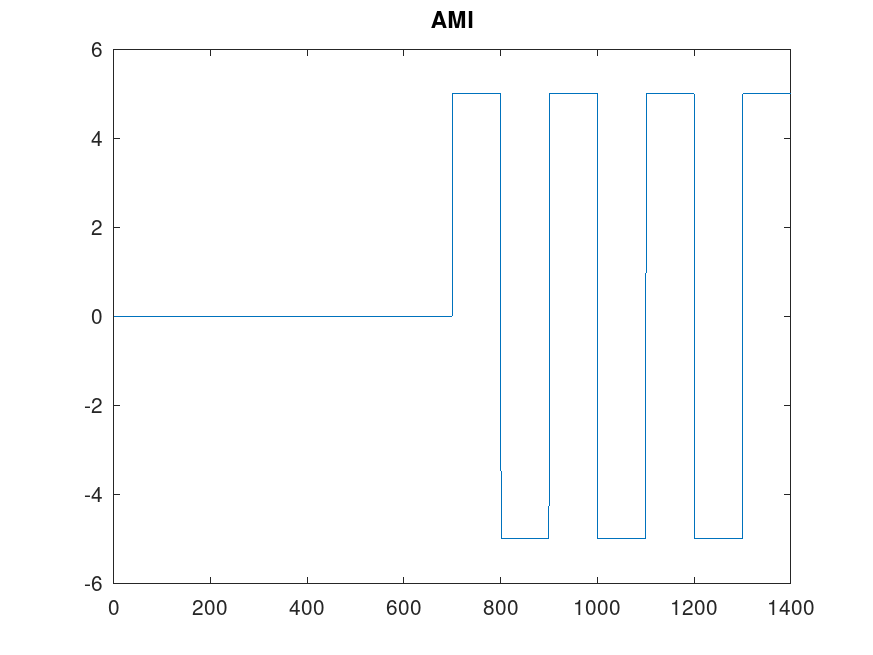
Манчестерское кодирование



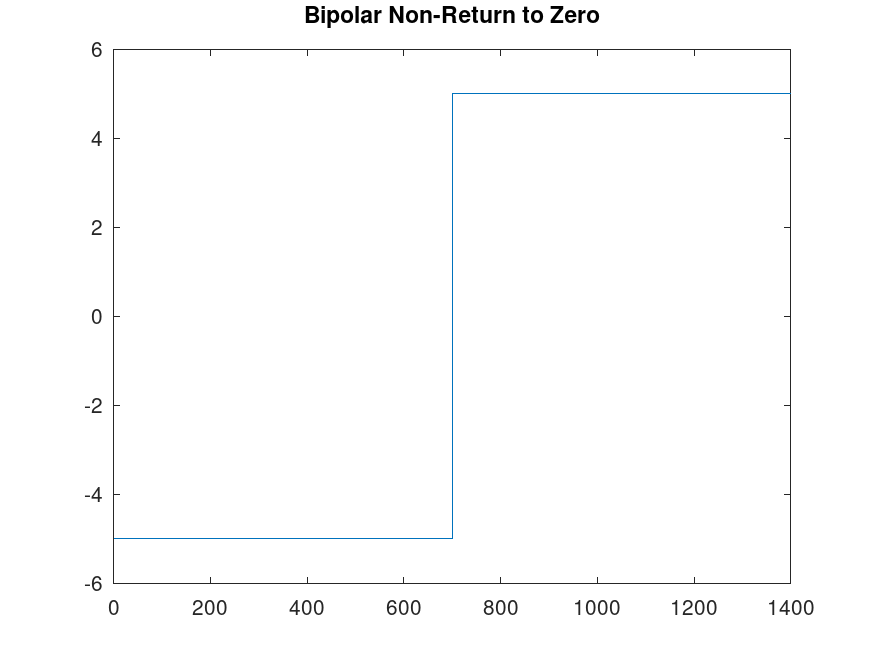
Дифференциальное манчестерское кодирование



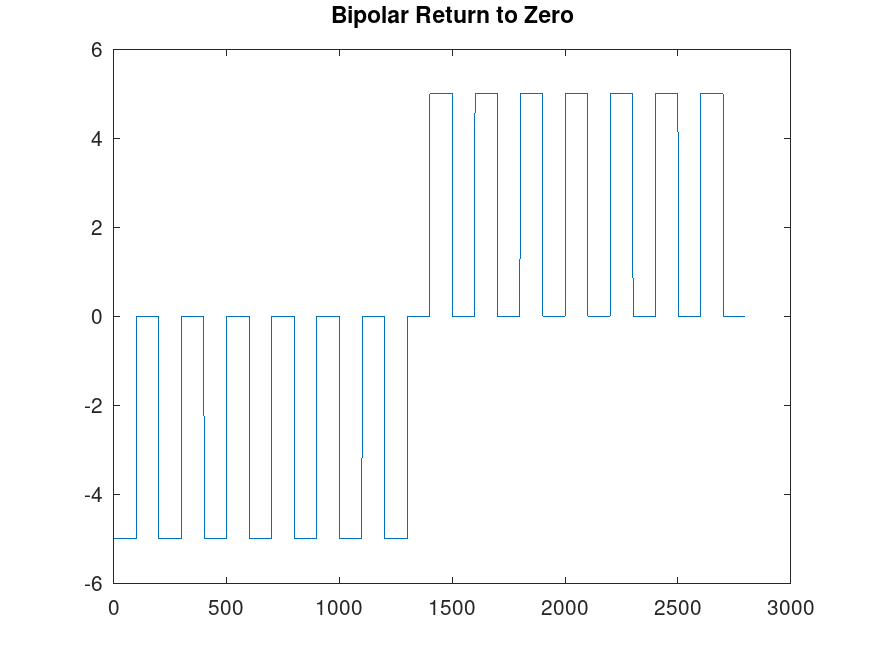
Униполярное кодирование: нет самосинхронизации



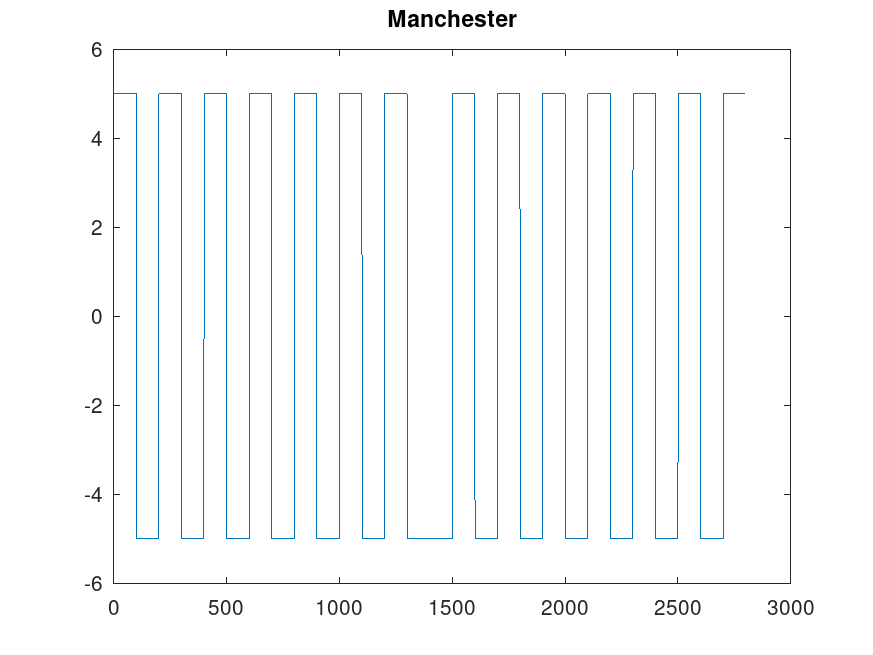
Кодирование AMI: самосинхронизация при наличии сигнала



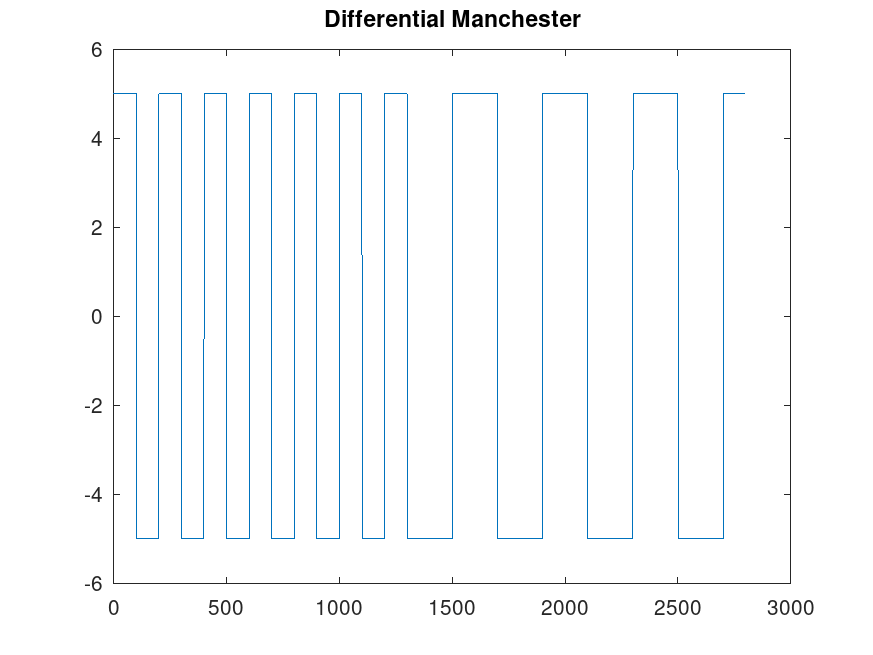
Кодирование NRZ: нет самосинхронизации



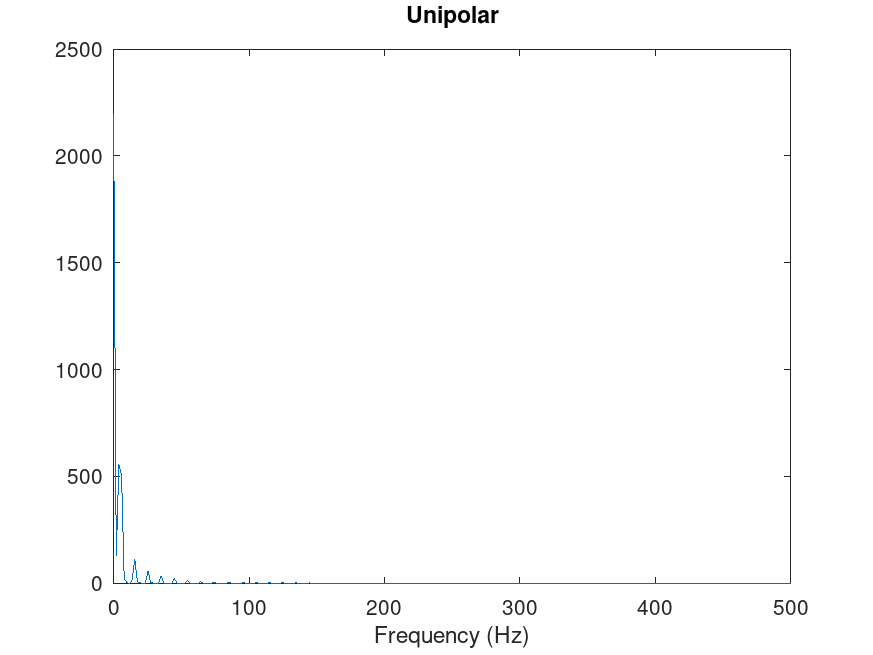
Кодирование RZ: есть самосинхронизация



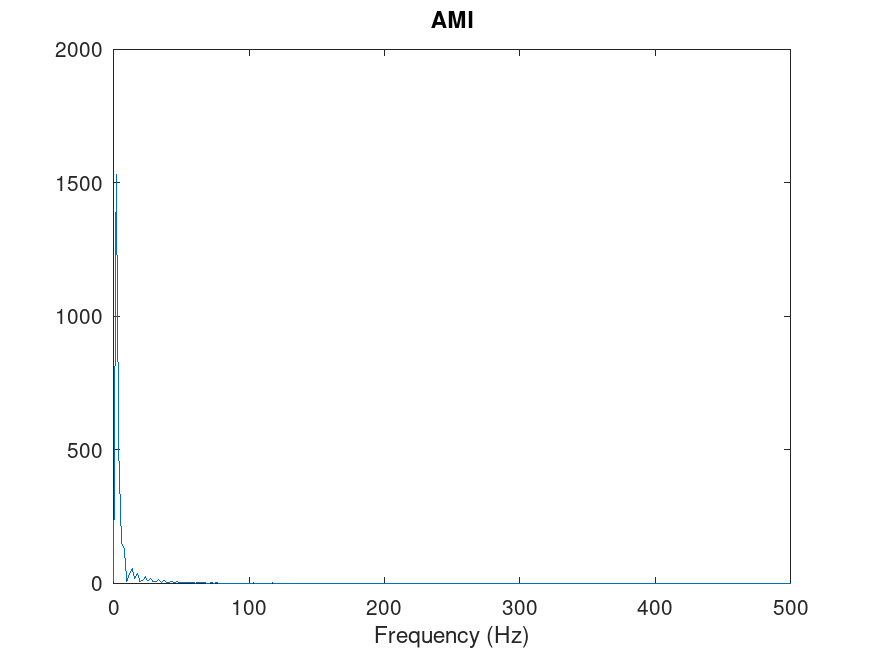
Манчестерское кодирование: есть самосинхронизация



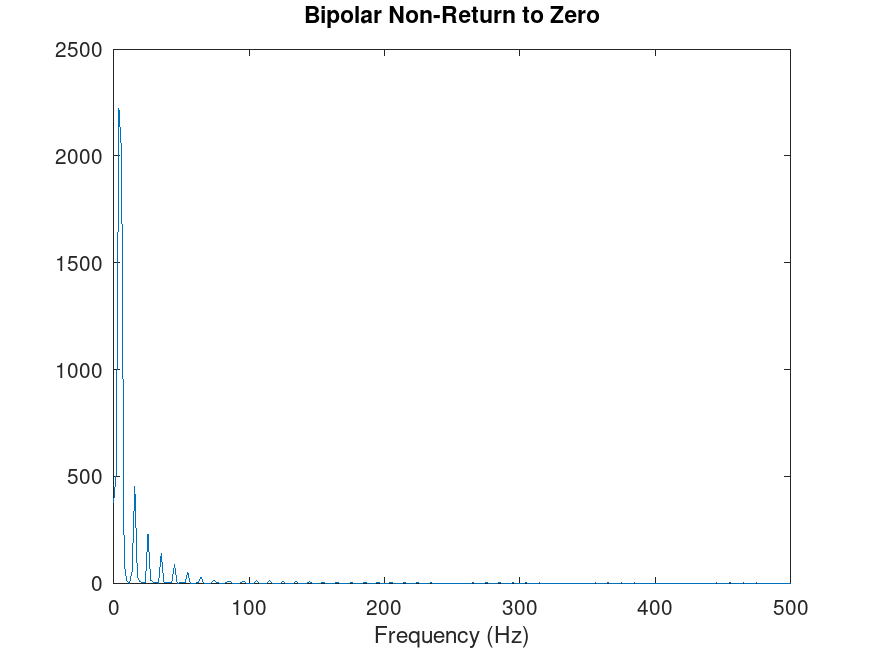
Дифференциальное манчестерское кодирование: есть самосинхронизация



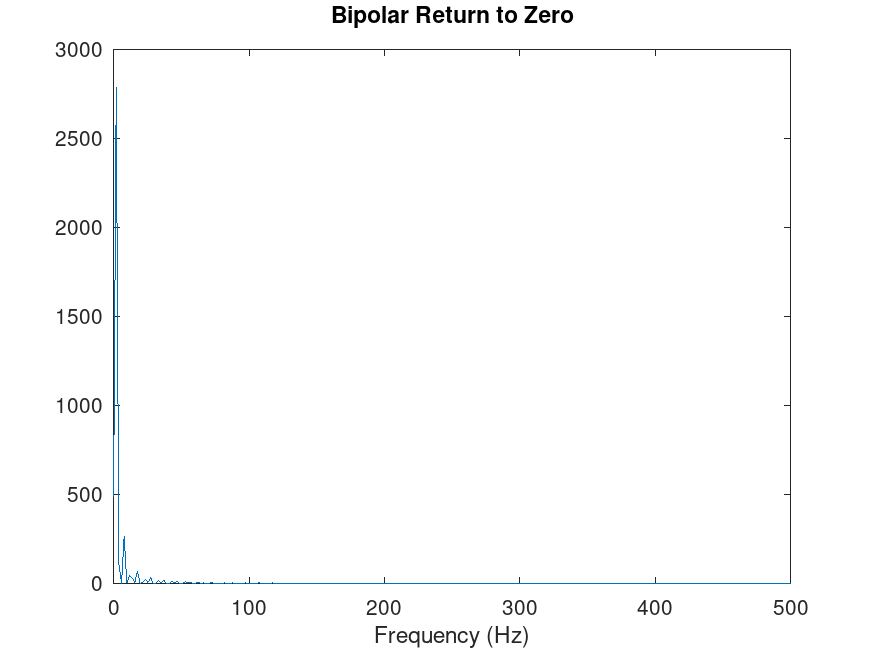
Униполярное кодирование: спектр сигнала



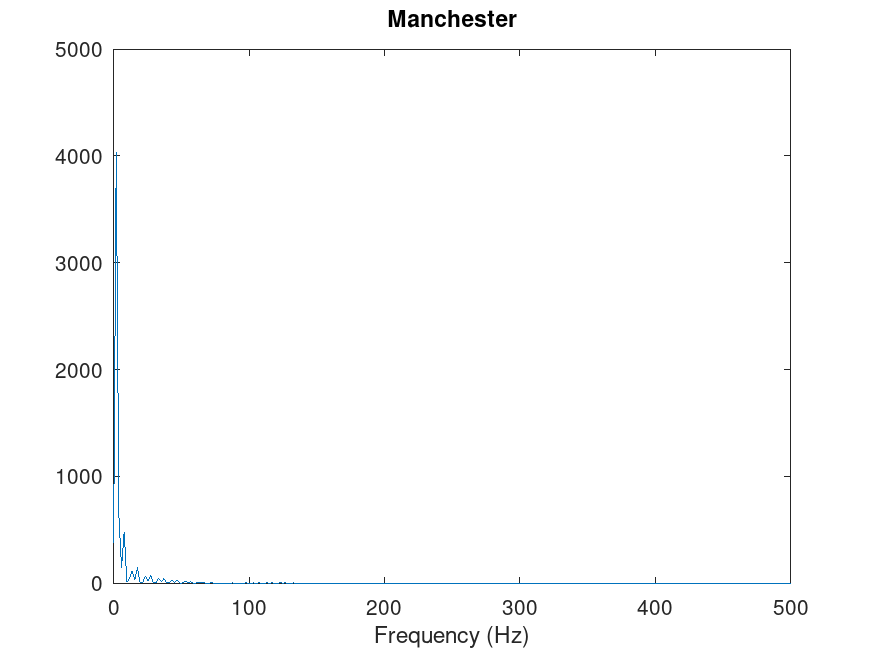
Кодирование AMI: спектр сигнала



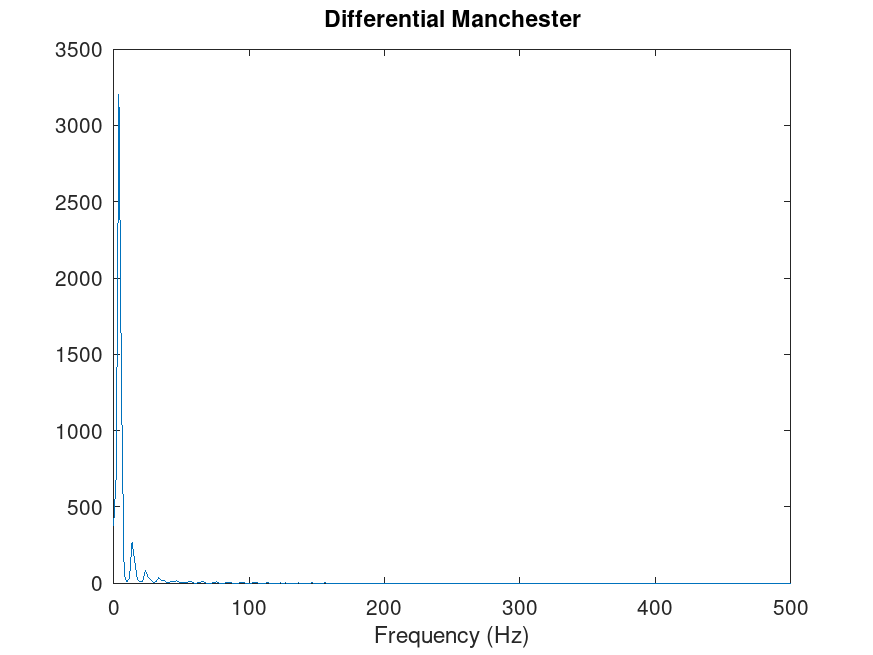
Кодирование NRZ: спектр сигнала



Кодирование RZ: спектр сигнала



Манчестерское кодирование: спектр сигнала



Дифференциальное манчестерское кодирование: спектр сигнала

# 3 Выводы

* Научились строить графики в Octave
* Реализовано азложение импульсного сигнала в форме меандра в частичный ряд Фурье через формулу как с синусами, так и с косинусами
* Показали,спектр суммы сигналов должен быть равен сумме спектров сигналов
* Получили, что спектр произведения представляет собой свёртку спектров
* Реализовали кодирование сигналов по битовым последовательностям
* Проверили свойства самосинхронизируемости кодов
* Получили спектры закодированных сигналов