Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого»
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Телекоммуникационные технологии

Лабораторная работа №3:
Линейная фильтрация

Работу выполнил:
Сергеев А.А.
Группа: 33531/2
Преподаватель:
Богач Н.В.

Санкт-Петербург
2019

Содержание

1 Цель работы

Изучить воздействие ФНЧ на тестовый сигнал с шумом.

2 Постановка задачи

Сгенерировать гармонический сигнал с шумом и синтезировать ФНЧ. Получить сигнал во временной и частотной областях до и после фильтрации. Сделать выводы о воздействии ФНЧ на спектр сигнала.

3 Теоретический раздел

3.1 Сущность линейной дискретной обработки

Линейный дискретный фильтр – это произвольная система обработки сигнала, обладающая свойствами линейности и стационарности.

- Линейность означает, что выходная реакция на сумму воздействий равна сумме реакций на эти воздействия, поданные на вход по отдельности.
- Стационарность означает, что задержка входного сигнала приводит лишь к такой же задержке выходного сигнала, не меняя его формы.

В общем случае дискретный фильтр суммирует (с весовыми коэффициентами) неко- торое количество входных отсчетов (включая последний) и некоторое количество предыдущих выходных отсчетов:

$$y(k) = b_0x(k) + b_1x(k-1) + b_2x(k-2) + ... + b_mx(k-m) - a_1y(k-1) - a_2y(k-2) - ... - a_ny(k-n)$$

где a_j, b_i – вещественные коэффициенты.
Вышеприведенная формула называется алгоритмом или уравнением дискретной фильтрации. Если сгруппировать слагаемые так, чтобы с одной строны от знака равенства были только входные отсчеты, а с другой – только выходные, получим форму записи, называемую разностным уравнением:

$$y(k) + a_1y(k-1) + a_2y(k-2) + ... + a_ny(k-n) = b_0x(k) + b_1x(k-1) + b_2x(k-2) + ... + b_mx(k-m)$$

3.2 Описание дискретных систем

В случае линейных систем с постоянными параметрами для анализа прохождения любого сигнала достаточно знать результат прохождения элементарного импульса в виде δ-функции. Для дискретных систем вместо δ-функции рассматривают единичную импульсную функцию $x_0(k)$:

$$x_0(k) = \begin{cases} 1, & \text{k=0} \\ 0, & \text{k}\neq 0 \end{cases}$$

Реакция дискретной системы на единичный импульс $x_0(k)$ называется импульсной характеристикой этой системы и обозначается $h(k)$.
Для физически реализуемой системы $h(k) = 0$ при $k < 0$, поэтому выходной сигнал дискретной линейной стационарной системы выражается через входной как

$$y(k) = \sum_{m=-\infty}^{k} x(m)h(k-m) = \sum_{m=0}^{\infty} x(k-m)h(m)$$

Это означает, что система при вычислении очередного отсчета может оперировать только прошлыми значениями входного сигнала и еще ничего не знает о будущих. С математической точки зрения данная формула представляет собой дискретную линейную свертку входного сигнала с импульсной характеристикой системы, через которую проходит этот сигнал.

3.3 Функция передачи

Свертке сигналов во временной области соответствует произведение их спектров в частотной. При анализе дискретных сигналов используют так называемое z-преобразование, свойства которого аналогичны преобразованиям Фурье и Лапласа. Поэтому дискретной свертке $x(k)$ и $h(k)$ будет соответствовать произведение z-преобразований этих последовательностей:

$Y(z) = H(z)X(z)$,
где $H(z)$ – функция передачи (transfer function) или системная функция:

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{\infty} h(k)z^{-k}$$

3.4 Частотная характеристика

Если подставить в функцию передачи $e^{j\omega T}$(здесь T – период дискретизации) в качестве аргумента, получим частотную характеристику дискретной системы:

$$\dot{K}(\omega) = H(e^{j\omega t}) = \sum_{k=0}^{\infty} h(k)e^{-j\omega kT}$$

3.5 Нерекурсивные фильтры

Нерекурсивными называют фильтры, которые не используют предыдущие отсчеты выходного сигнала при вычислении следующих. Для них уравнение фильтрации упрощается и принимает вид:

$$y(k) = \sum_{i=0} Kb_i x(k-i)$$

Количество K используемых отсчетов входного сигнала называется порядком фильтра. Структурная схема, реализующая алгоритм, приведена на рис. 1. Блоки, подписанные как «z^{-1}», соответствуют задержке входного сигнала на 1 такт (что равносильно умножению z-преобразования задерживаемой последовательности на z^{-1}).

Импульсная характеристика нерекурсивного фильтра определяется путем подстановки в уравнение единичного импульса $x_0(k)$ в качестве входного сигнала:

$$h(k) = \sum_{i=0} Kb_i x_0(k-i) = b_k$$

так как $x_0(k-i) \neq 0$ только для $i = k$.

Таким образом, коэффициенты b_i нерекурсивного фильтра являются отсчетами его импульсной характеристики. Так как в реальном устройстве линия задержки содержит конечное число элементов, то импульсная характеристика нерекурсивного фильтра также является конечной по длительности.
Поэтому еще одно название таких фильтров – фильтры с конечной импульсной характеристикой (КИХ-фильтры, английский термин – finite impulse response, FIR).

3.6 Рекурсивные фильтры

Если уравнение фильтрации содержит как входные, так и выходные отсчеты, то для реализации такого фильтра необходимо добавить вторую линию задержки – для хранения выходных отсчетов $y(k-i)$.

Так как при вычислениях используются предыдущие отсчеты выходного сигнала, в схеме присутствуют обратные связи. Поэтому такие фильтры называют рекурсивными. Для таких фильтров уравнение фильтрации может быть записано следующим образом:

$$y(k) = \sum_{i=0} Kb_i x(k-i) - \sum_{i=1} Na_i y(k-i)$$

где $max(K, N)$ называют порядком данного рекурсивного фильтра.

Так как каждый следующий отсчет импульсной характеристики рекурсивного фильтра вычисляется исходя из значения предыдущего, то количество таких отсчетов не ограничено. Поэтому рекурсивные фильтры называют также фильтрами с бесконечной импульсной характеристикой (БИХфильтрами, английский термин – infinite impulse response, IIR). По этой же причине рекурсивные фильтры могут быть неустойчивыми.

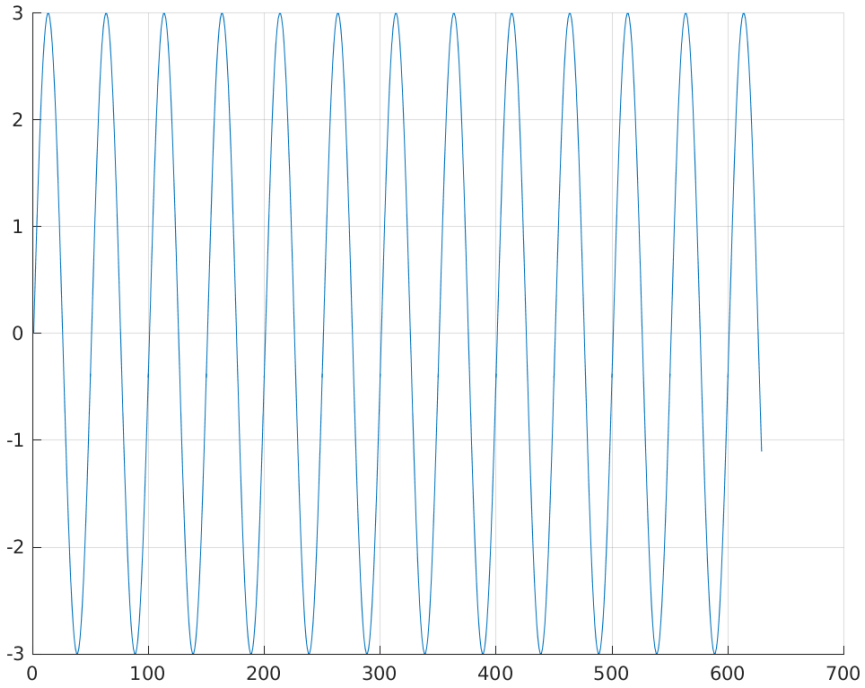
4 Ход работы

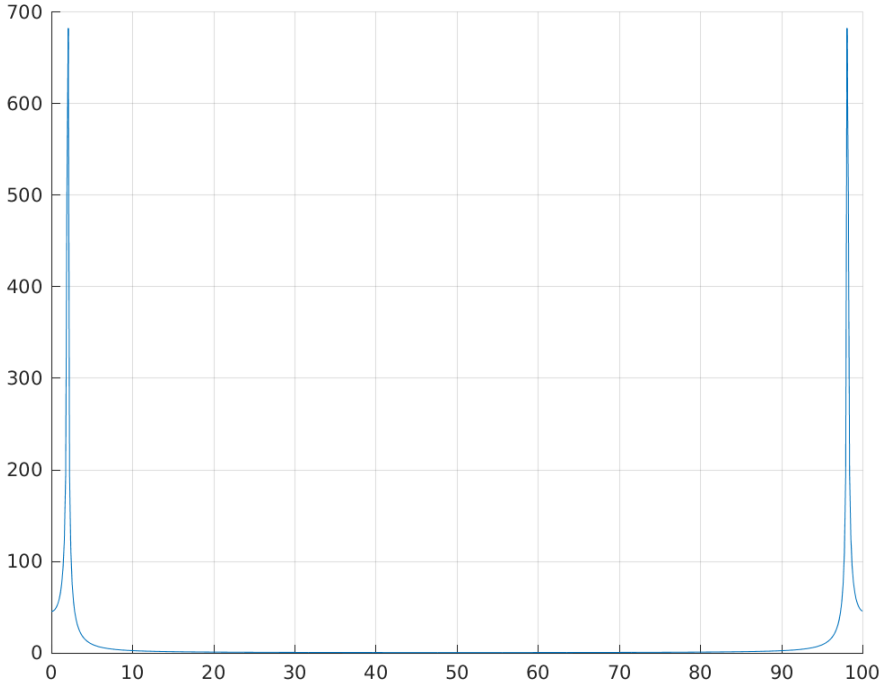
Для демонстрации работы фильтров используем сигналы из лабораторной работы 1.

4.1 Синусоидальный сигнал

4.1.1 Синусоидальынй сигнал и его спектр

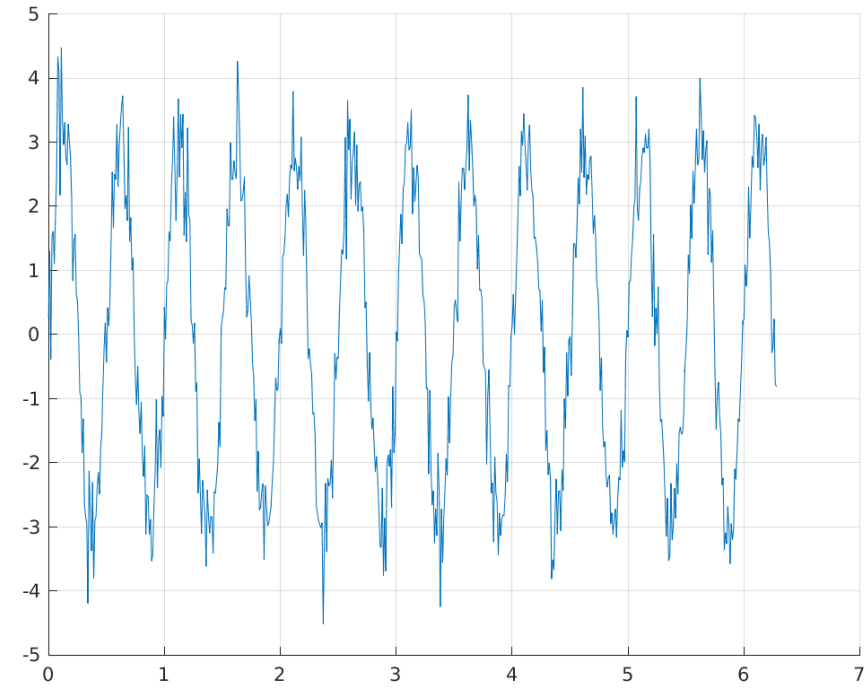
```
function [y] = sin_sig
    global t tau fs;

    a    = 3;   %амплитуда
    fc   = 2;   %частота

    %синусоидальный сигнал
    y = a * sin(2 * pi * fc * t);
    draw_img(y);

    %спектр синусоидального сигнала
    sp     = abs(fft(y));
    fplot = 0:1\tau:fs;
    draw_img(fplot, sp);
end
```

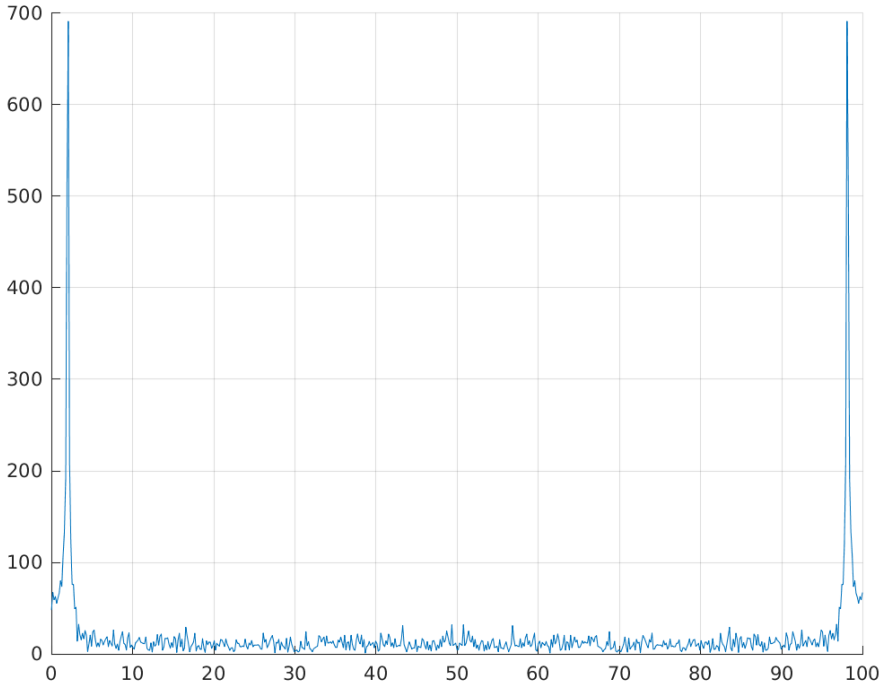




4.1.2 Зашумленный синусоидальный сигнал и его спектр

```
function [x_noise, sp, fplot] = noise_sin_sig(y)
    global t tau fs;
    %зашумленный синусоидальный сигнал
    x_noise = awgn(y, 6);
    draw_img(t, x_noise);

    %спектр зашумленного синусоидального сигнала
    sp    = abs(fft(x_noise));
    fplot = 0:1/tau:fs;
    draw_img(fplot, sp);
end
```





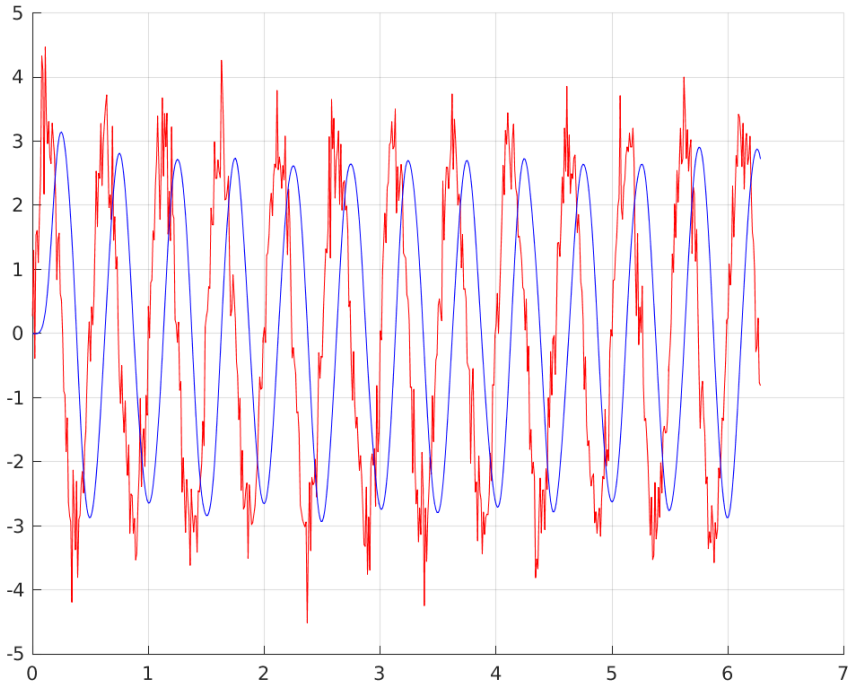
4.1.3 Применение фильтра

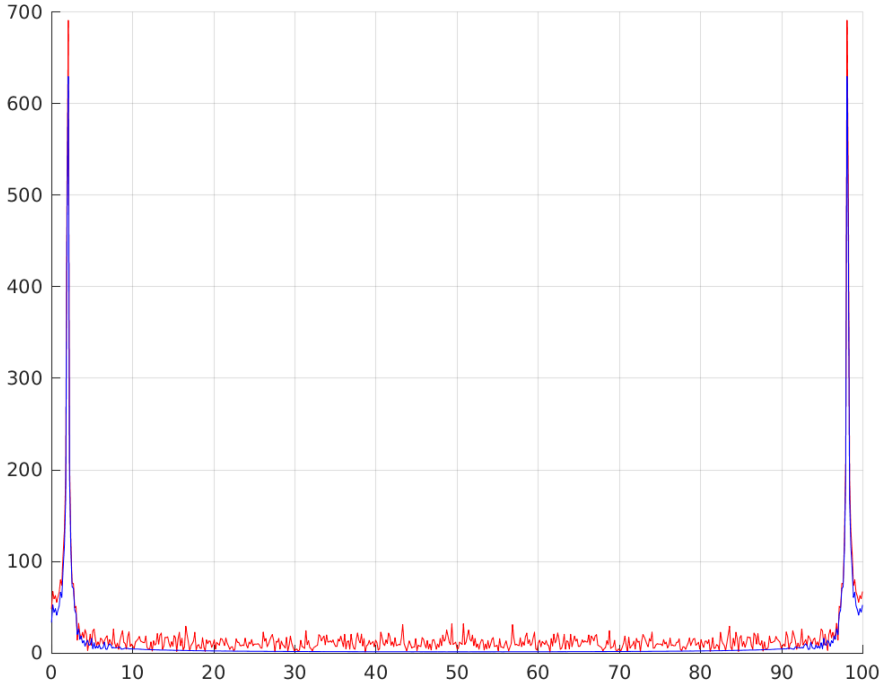
Релизация фильтра и его настройка:

```
function [Hd] = filter
    Fs = 20;
    Fpass = 0;
    Fstop = 2;
    Dpass = 0.007501127785;
    Dstop = 3;
    flag  = 'scale';

    [N,Wn,BETA,TYPE] = kaiserord([Fpass Fstop]/(Fs/2), [1 0], [Dstop Dpass]);

    b  = fir1(N, Wn, TYPE, kaiser(N+1, BETA), flag);
    Hd = dfilt.dffir(b);
end
```

Фильтрация сигнала:

```
function filter_sin_sig(x_noise, sp, fplot)
    global t Hd;
    %синусоидальный сигнал после фильтрации
    filtered_signal = filter(Hd, x_noise);
    draw_img(t, x_noise, 'r', t, filtered_signal, 'b');

    %спектр синусоидального сигнала после фильтрации
    sp_filter = abs(fft(filtered_signal));
    draw_img(fplot, sp(1:length(fplot)), 'r', fplot, sp_filter(1:length(fplot)), 'b');
end
```

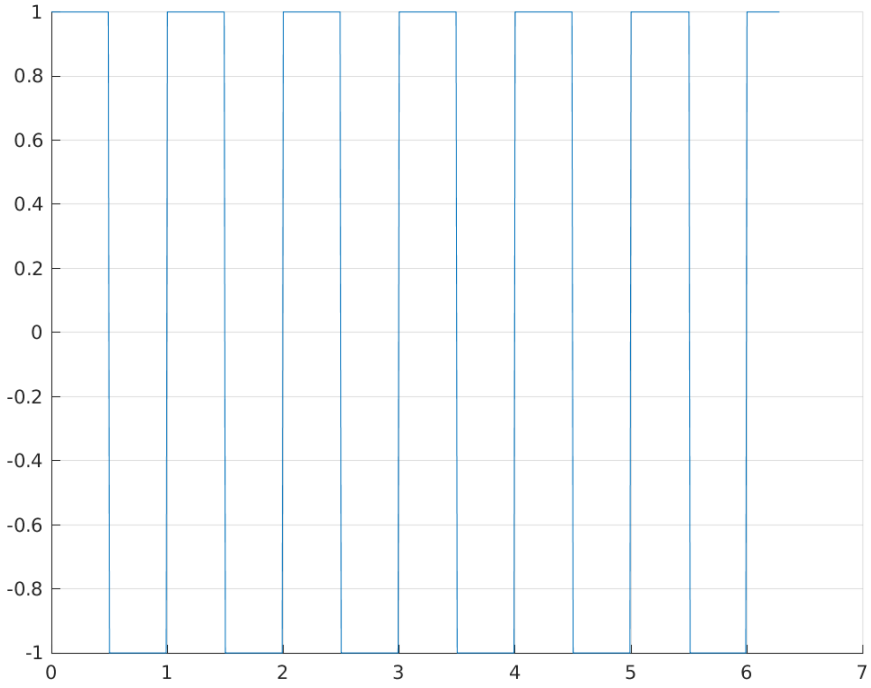


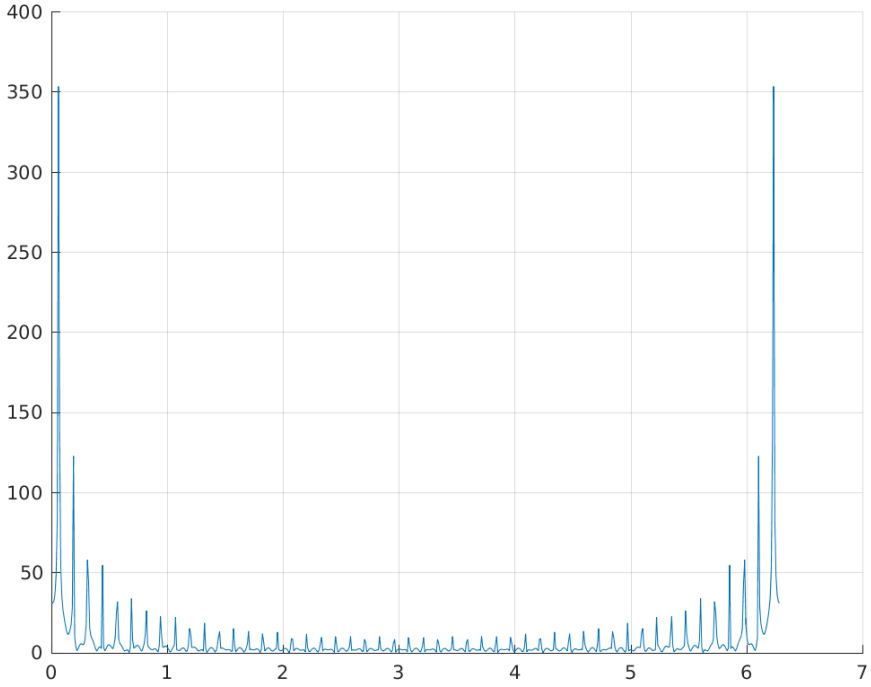


4.2 Прямоугольный сигнал

4.2.1 Прямоугольный сигнал и его спектр

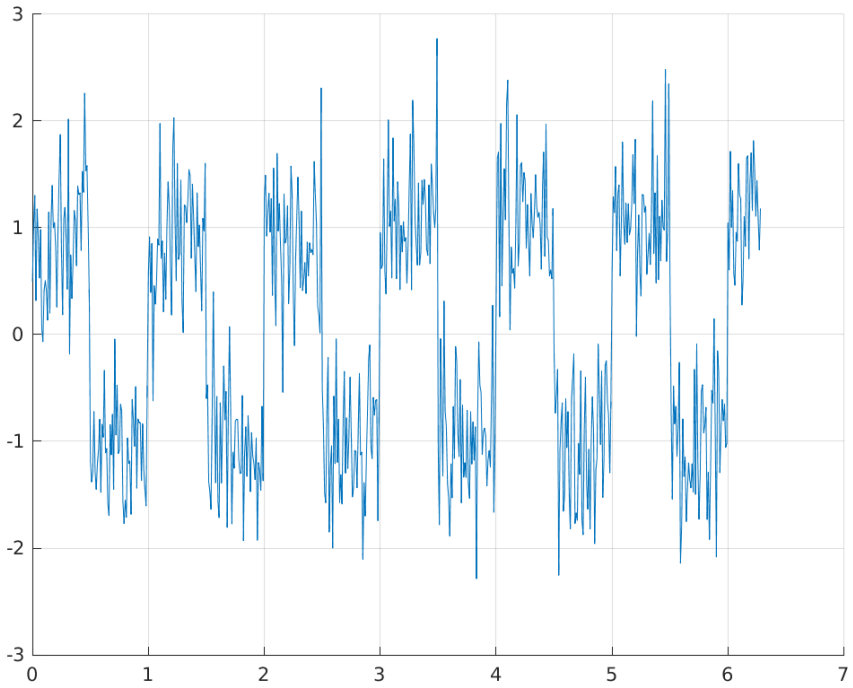
```
function [x] = square_sig
    global t;
    %прямоугольный сигнал
    duty = 50;
    amp  = 1;
    x    = amp * square(2 * pi * t, duty);
    draw_img(t, x);

    %спектр прямоугольного сигнала
    sp = abs(fft(x));
    draw_img(t, sp);
end
```

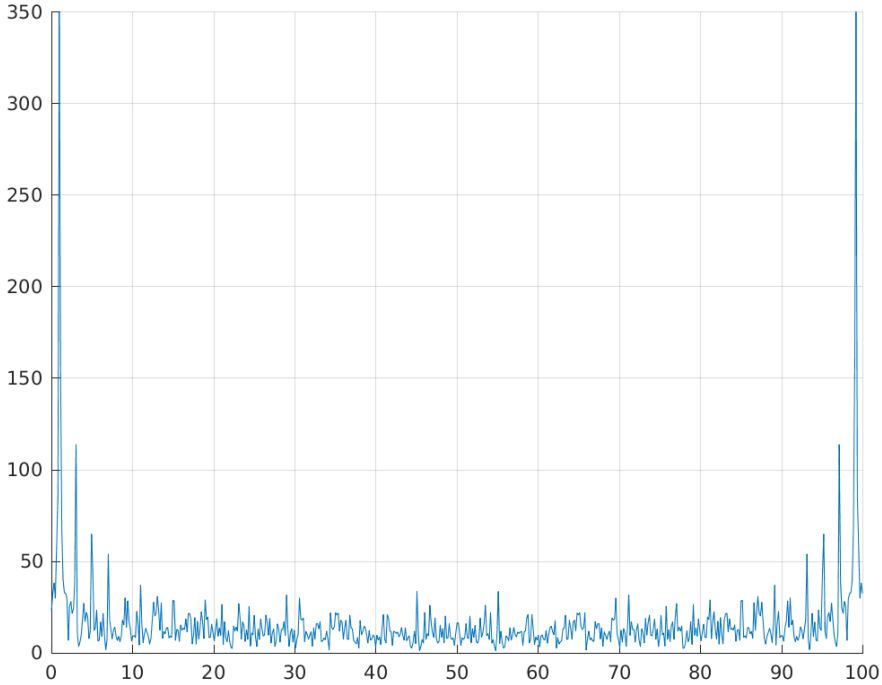




4.2.2 Зашумленный прямоугольный сигнал и его спектр

```
function [x_noise, sp, fplot] = noise_square_sig(x)
    global t tau fs;
    %зашумленный прямоугольный сигнал
    x_noise = awgn(x, 6);
    draw_img(t, x_noise);

    %спектр зашумленного прямоугольного сигнала
    sp    = abs(fft(x_noise));
    fplot = 0:1/tau:fs;
    draw_img(fplot, sp);
end
```





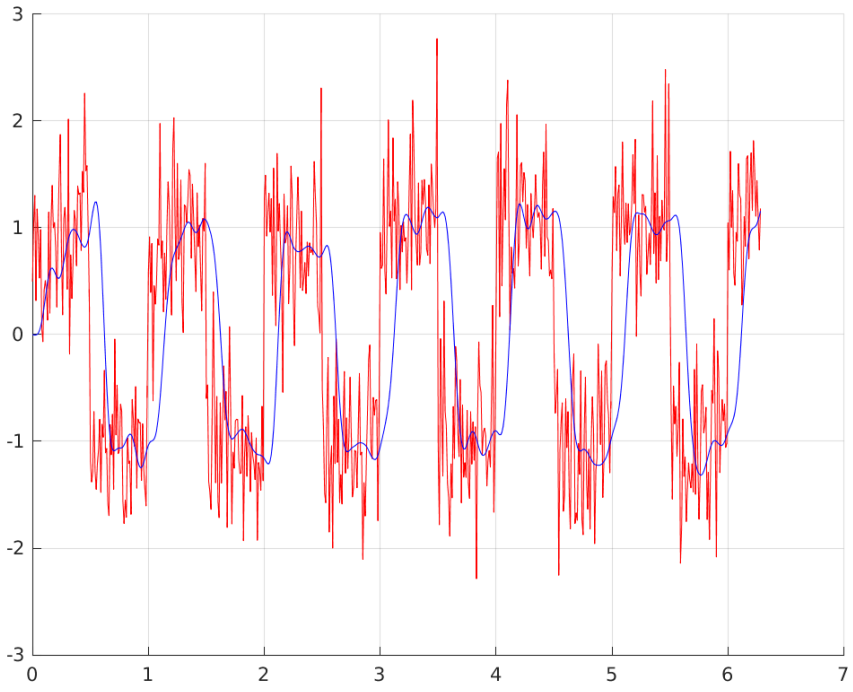
4.2.3 Применение фильтра

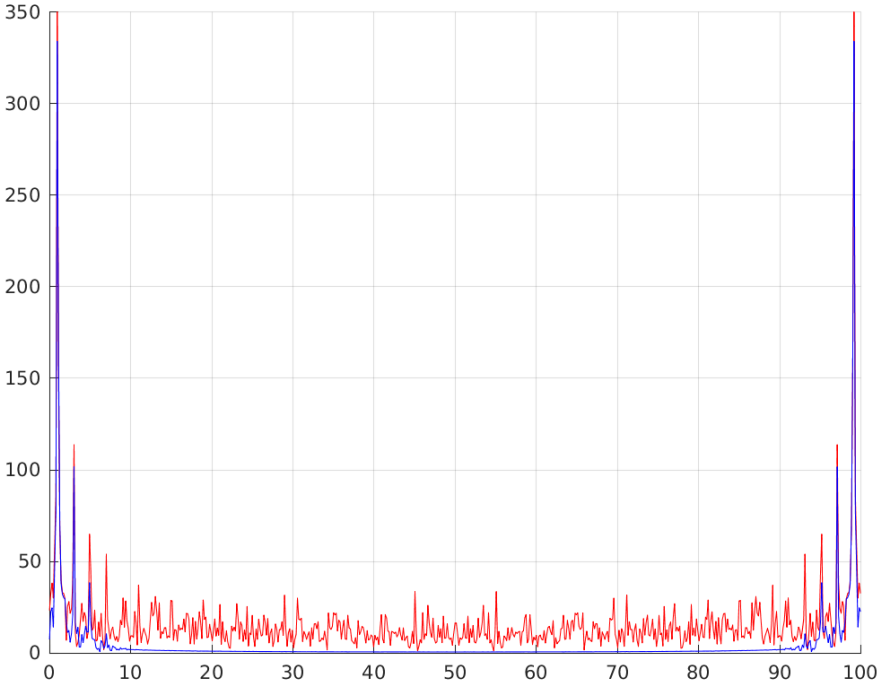
Релизация фильтра и его настройка:

```
function [Hd] = filter
    Fs = 20;
    Fpass = 0;
    Fstop = 2;
    Dpass = 0.007501127785;
    Dstop = 3;
    flag  = 'scale';

    [N,Wn,BETA,TYPE] = kaiserord([Fpass Fstop]/(Fs/2), [1 0], [Dstop Dpass]);

    b  = fir1(N, Wn, TYPE, kaiser(N+1, BETA), flag);
    Hd = dfilt.dffir(b);
end
```

Фильтрация сигнала:

```
function filter_square_sig(x_noise, sp, fplot)
    global Hd t;
    %прямоугольный сигнал после фильтрации
    filtered_signal = filter(Hd, x_noise);
    draw_img(t, x_noise, 'r', t, filtered_signal, 'b');

    %спектр прямоугольного сигнала после фильтрации
    sp_filter = abs(fft(filtered_signal));
    draw_img(fplot, sp(1:length(fplot)), 'r', fplot, sp_filter(1:length(fplot)), 'b');
end
```

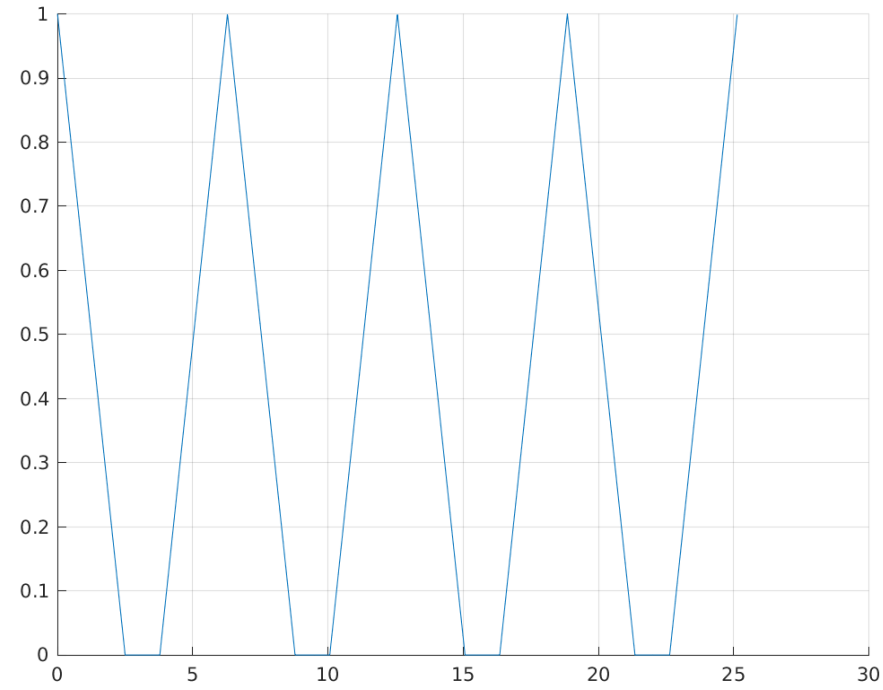


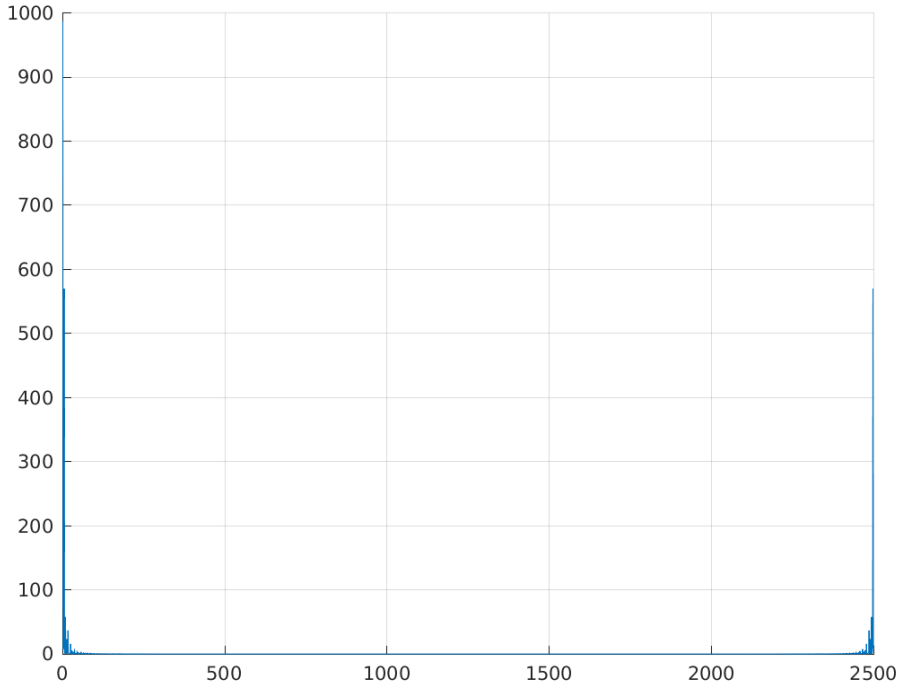


4.3 Треугольный сигнал

4.3.1 Треугольыний сигнал и его спектр

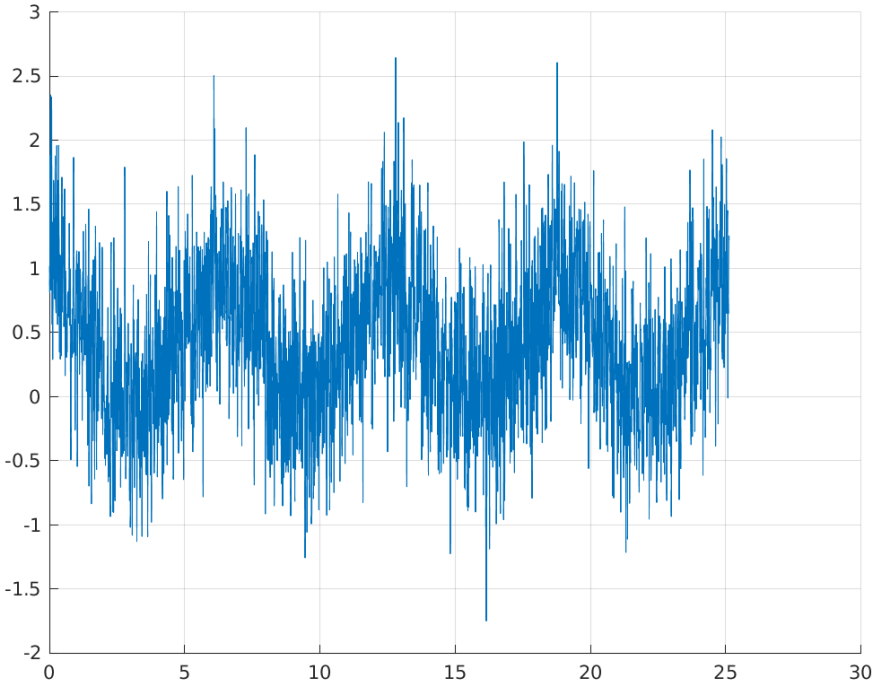
```
function [s] = triangle_sig
    global t tau fs;
    %треугольный сигнал
    T = tau ;
    tau = 5;
    fs  = 1/100;
    N = 5;
    t   = 0:fs:(N-1)*T; %время
    d = (0:(N-1))*T;
    A = 1;
    s = A * pulstran(t, d, 'tripuls', tau);
    draw_img(t, s);
    sp = fft(s, 2500);
    draw_img(abs(sp));
end
```

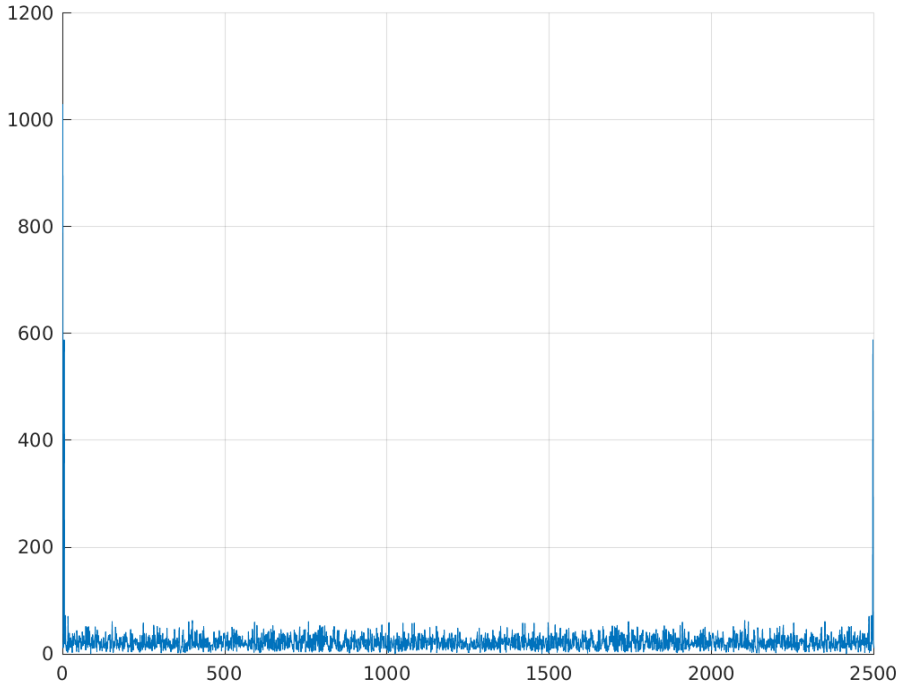




4.3.2 Зашумленный треугольный сигнал и его спектр

```
function [x_noise] = noise_triangle_sig(s)
    global t;
    %зашумленный треугольынй сигнал
    x_noise = awgn(s, 6);
    draw_img(t, x_noise);

    %спектр зашумленного треугольного сигнала
    draw_img(abs(fft(x_noise, 2500)));
end
```





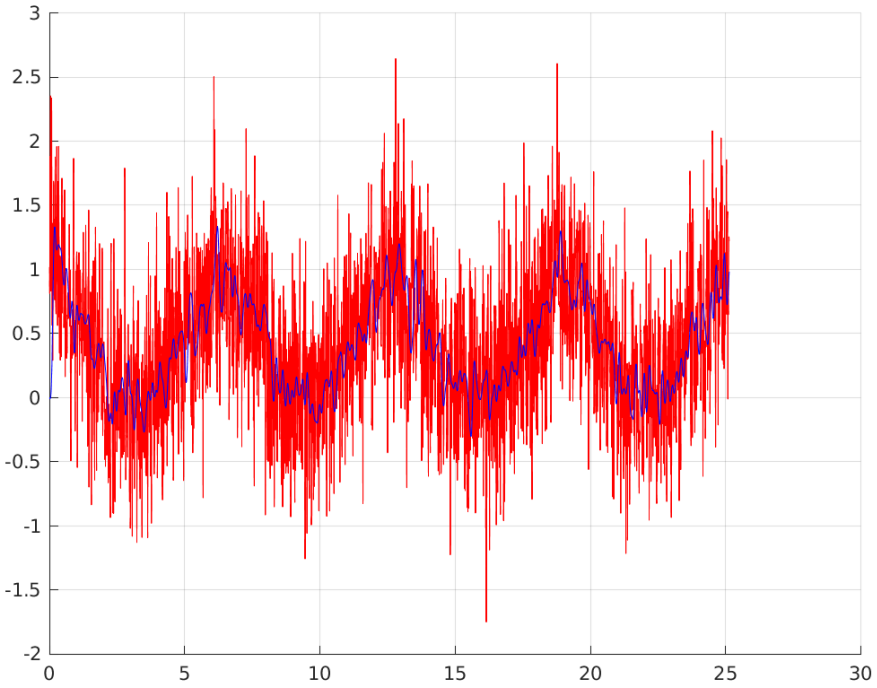
4.3.3 Применение фильтра

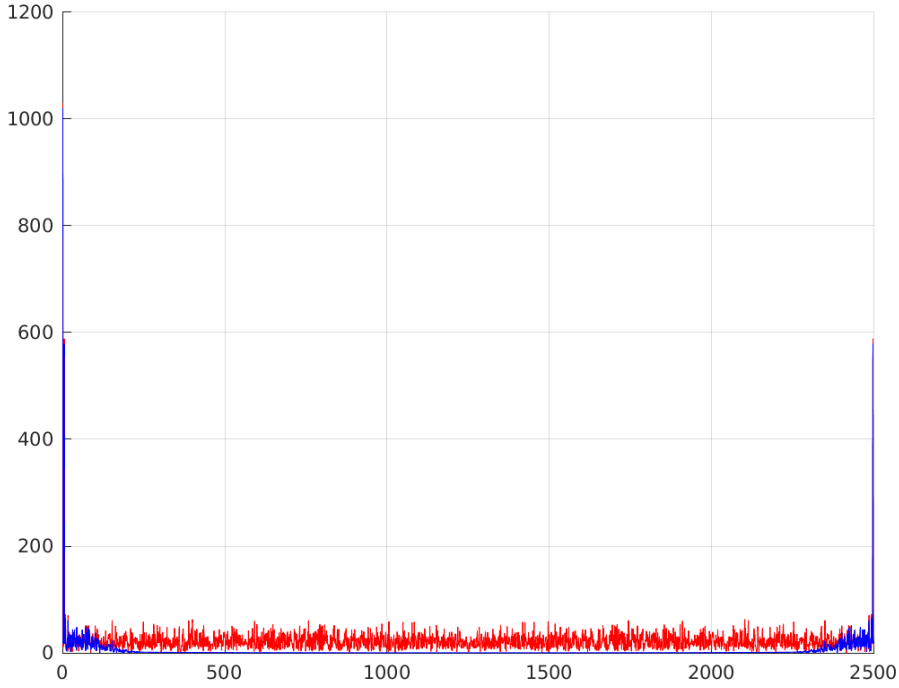
Релизация фильтра и его настройка:

```
function [Hd] = filter
    Fs = 20;
    Fpass = 0;
    Fstop = 2;
    Dpass = 0.007501127785;
    Dstop = 3;
    flag  = 'scale';

    [N,Wn,BETA,TYPE] = kaiserord([Fpass Fstop]/(Fs/2), [1 0], [Dstop Dpass]);

    b  = fir1(N, Wn, TYPE, kaiser(N+1, BETA), flag);
    Hd = dfilt.dffir(b);
end
```

Фильтрация сигнала:

```
function filter_triangle_sig(x_noise)
    global Hd t;
    %треугольный сигнал после фильтрации
    filtered_signal = filter(Hd, x_noise);
    draw_img(t, x_noise, 'r', t, filtered_signal, 'b');

    %спектр треугольного сигнала после фильтрации
    sp_filter = abs(fft(filtered_signal, 2500));
    draw_img(1:2500, abs(fft(x_noise, 2500)), 'r', 1:2500, sp_filter, 'b');
end
```





5 Вывод

В ходе данной работы было рассмотрено восстановление зашумленных сигналов с помощью линейного фильтра. В результате фильтрации часть высокочастотных шумов исчезла. В результате на выходе мы наблюдали частично восстановленный гармонический сигнал. Причина неполного удаления шума линейным фильтром в том, что белый шум имеет сплошной спектр, из чего следует, что он попадет в полосу пропускания фильтра. Из-за этого мы наблюдаем на выходе гармонический сигнал не полностью восстановленный, а слегка с искажениями.