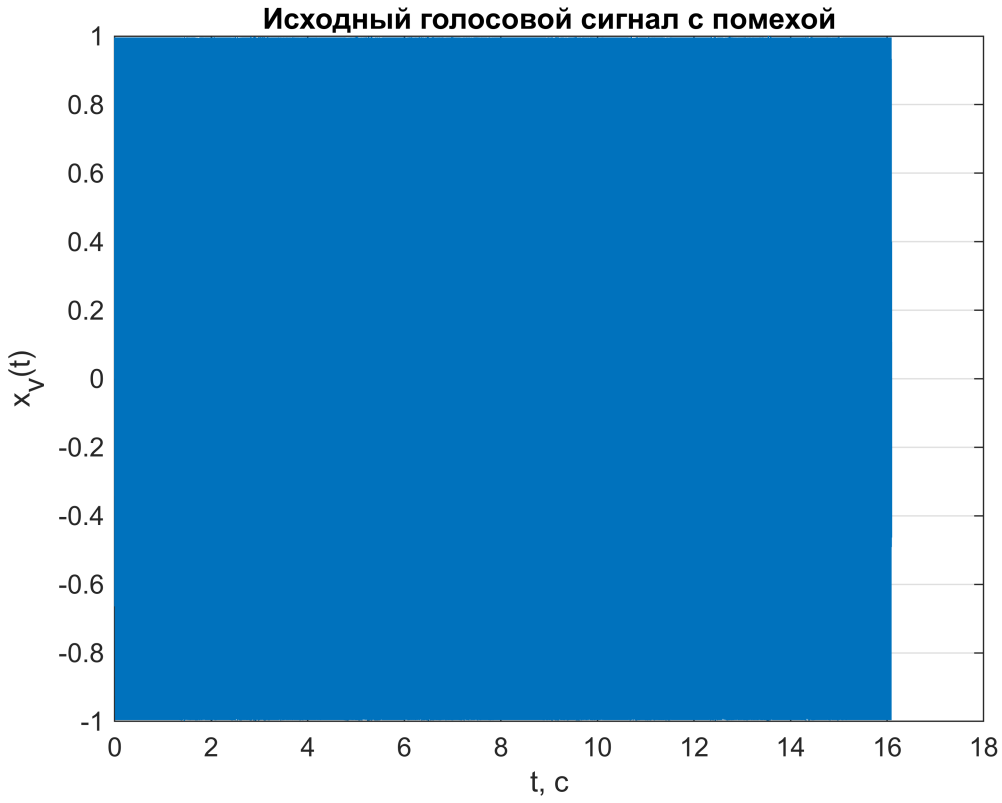
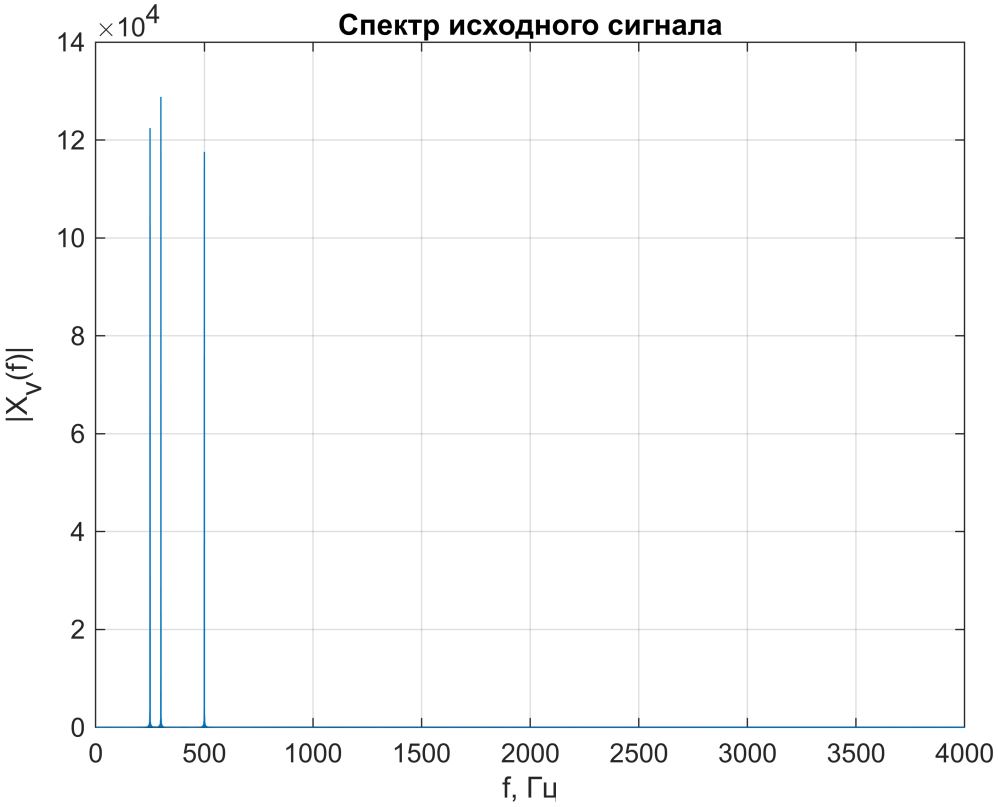
```
[xv, fs_v] = audioread('W4_07.wav');
xv = xv(:,1); % берём только первый канал
t_v = (0:length(xv)-1)/fs_v; % формируем временную ось для голосового сигнала

figure;
plot(t_v, xv);
xlabel('t, c');
ylabel('x_v(t)');
title('Исходный голосовой сигнал с помехой');
grid on;
```



Спектр исходного сигнала

```
N_v = 2^nextpow2(length(xv)); % длина ДПФ (степень двойки)
Xv = fft(xv, N_v); % спектр голосового сигнала
f_v = (0:N_v-1)*(fs_v/N_v); % ось частот

figure;
plot(f_v(1:N_v/2), abs(Xv(1:N_v/2)));
xlabel('f, Гц');
ylabel('|X_v(f)|');
title('Спектр исходного сигнала');
grid on;
xlim([0 4000]);
```

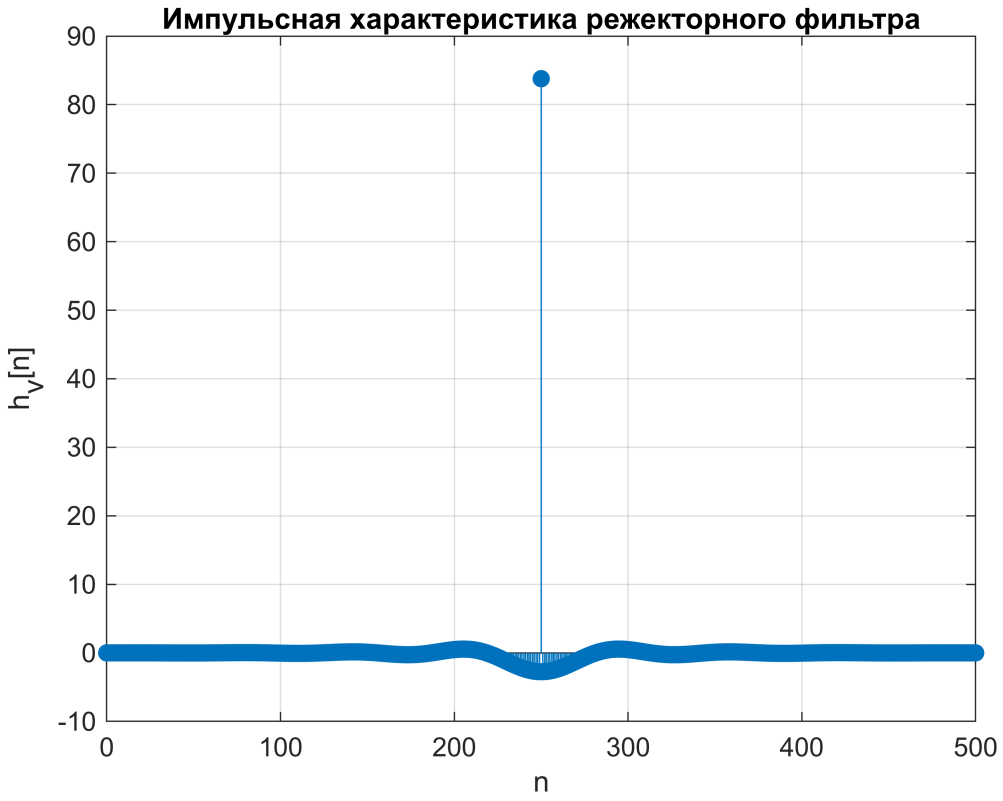


Параметры режекторного КИХ фильтра

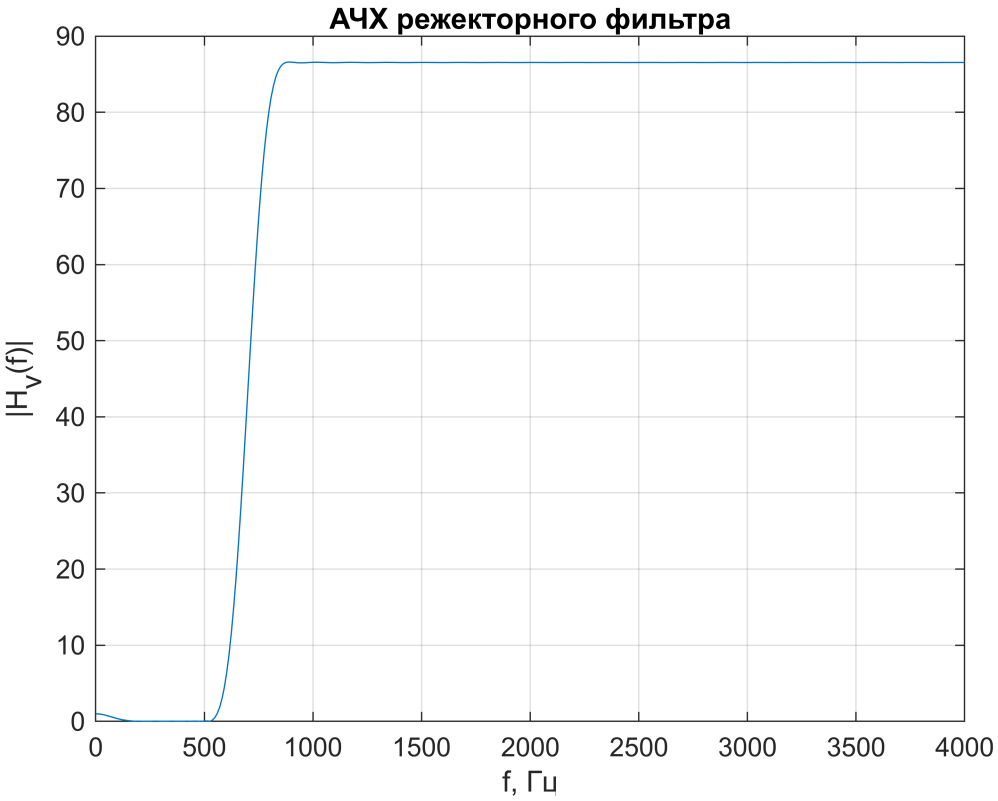
```
f1 = 1; % нижняя граница полосы заграждения, Гц
f2 = 700; % верхняя граница полосы заграждения, Гц

% Нормировка частот к частоте Найквиста (fs_v/2) по формуле W = f/(fs/2)
F1 = f1 / (fs_v/2); % нормированная нижняя граница полосы заграждения
F2 = f2 / (fs_v/2); % нормированная верхняя граница полосы заграждения

% Порядок фильтра и окно
M = 500; % порядок FIR-фильтра (длина ИХ = M+1 отсчёт)
beta = 6; % параметр окна Кайзера (определяет форму/ширину главного лепестка)

% Строим полосозаграждающий (режекторный) КИХ фильтр.
% Создаёт фильтр с полосой заграждения [F1 F2].
hv = fir1(M, [F1 F2], 'stop', kaiser(M+1, beta));
```

ИХ и АЧХ фильтра

```
figure;
stem(0:length(hv)-1, hv, 'filled'); % отображаем импульсную характеристику фильтра
xlabel('n');
ylabel('h_v[n]');
title('Импульсная характеристика режекторного фильтра');
grid on;
```




```
% Частотная характеристика режекторного фильтра
[Hv, wv] = freqz(hv, 1, 2048, fs_v);
figure;
plot(wv, abs(Hv));
xlabel('f, Гц');
ylabel('|H_v(f)|');
title('АЧХ режекторного фильтра');
grid on;
xlim([0 4000]);
```



Фильтрация сигнала

```
% 1-й проход: пропускаем сигнал через режекторный фильтр
y1 = filter(hv, 1, xv);

% 2-й проход тем же фильтром: дополнительно усиливаем подавление помехи
y2 = filter(hv, 1, y1);
```

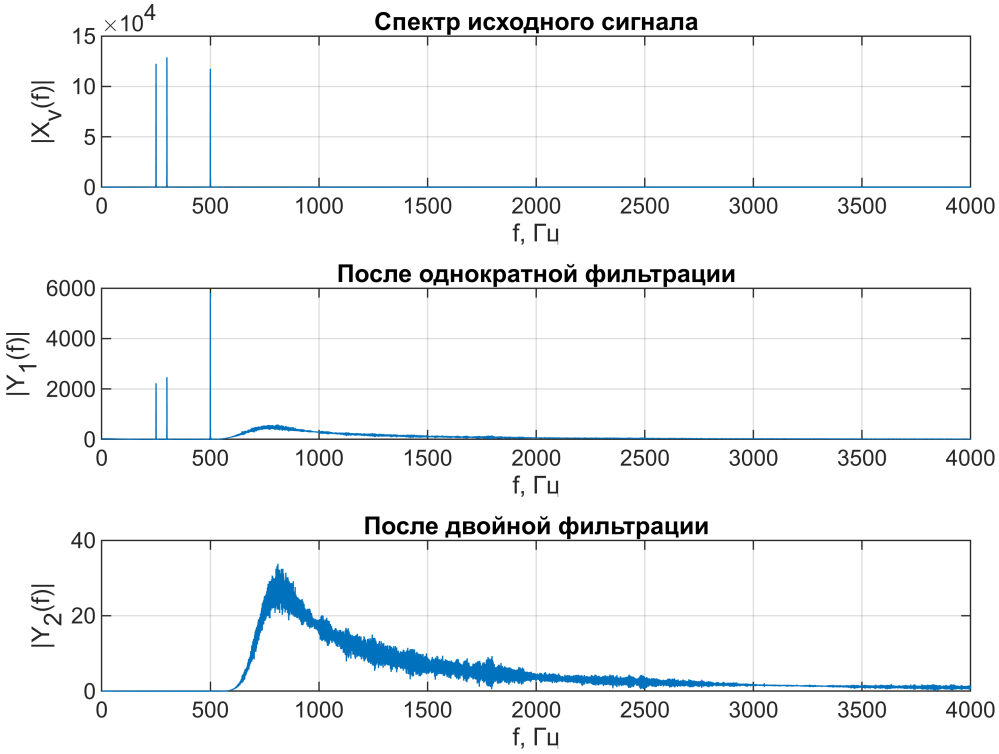
Нормализация, чтобы не было clipping при записи

```
% Находим максимальное по модулю значение в сигнале y2
maxVal = max(abs(y2));
if maxVal > 0
    % Делим весь сигнал на этот максимум, чтобы амплитуда была в диапазоне [-1, 1]
    % Это предотвращает обрезание (clipping) при воспроизведении и записи в файл.
    y2 = y2 / maxVal;
end
```

Спектры до/после

```
% Строим спектры исходного и отфильтрованных сигналов для сравнения
Y0 = fft(xv, N_v); % спектр исходного сигнала
Y1 = fft(y1, N_v); % спектр после одной фильтрации
Y2 = fft(y2, N_v); % спектр после двух последовательных фильтраций
```

```
figure;
subplot(3,1,1);
plot(f_v(1:N_v/2), abs(Y0(1:N_v/2)));
title('Спектр исходного сигнала');
grid on;
xlabel('f, Гц');
ylabel('|X_v(f)|');
xlim([0 4000]);

subplot(3,1,2);
plot(f_v(1:N_v/2), abs(Y1(1:N_v/2)));
title('После однократной фильтрации');
grid on;
xlabel('f, Гц');
ylabel('|Y_1(f)|');
xlim([0 4000]);

subplot(3,1,3);
plot(f_v(1:N_v/2), abs(Y2(1:N_v/2)));
title('После двойной фильтрации');
grid on;
xlabel('f, Гц');
ylabel('|Y_2(f)|');
xlim([0 4000]);
```



Сохранение результата

```
audiowrite('W4_07_filtered.wav', y2, fs_v);
```