Изучение тембра и синтез его математической модели

1. Извлечение гармоник из спектра

Красницкий Никита

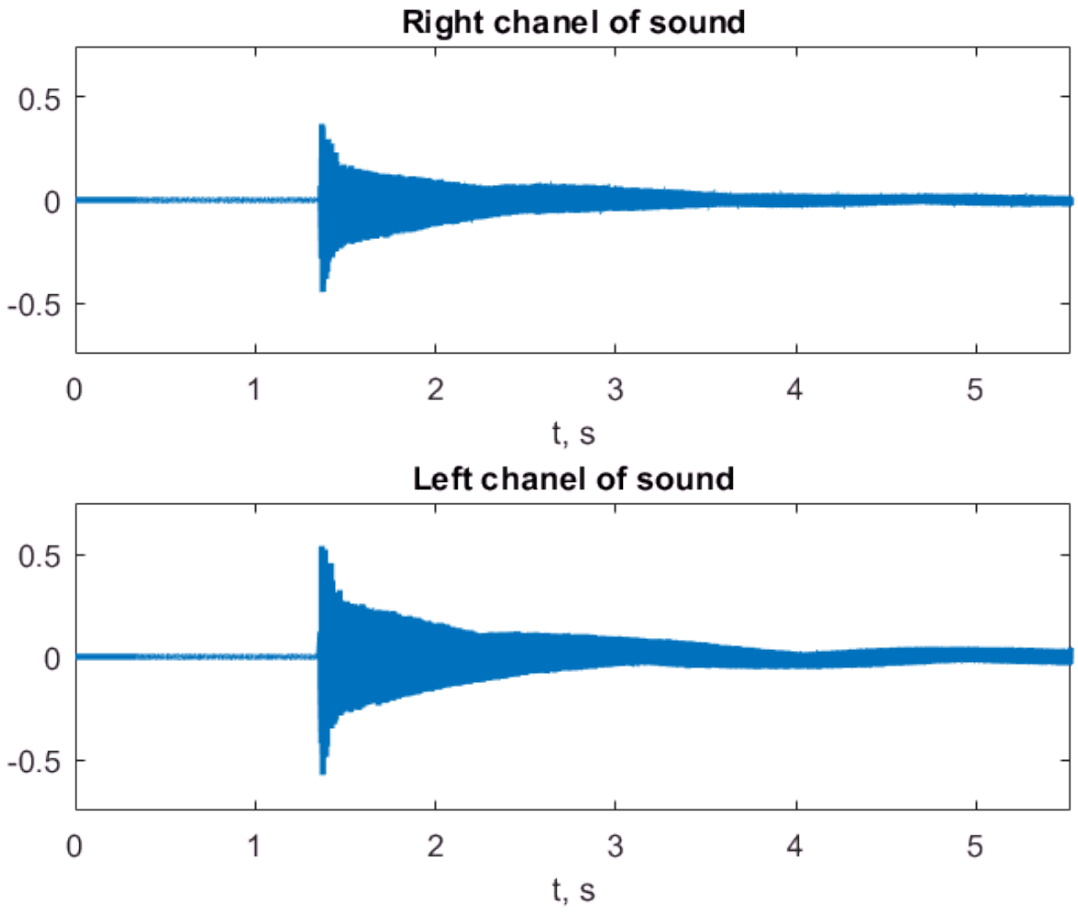
Тембр - есть окрас музыкального заука, который может быть описан отношением амплитуд гармоник звука и их временными зависимостями. Для исследования тембра возьму в пример извлеченный звук гитарной струны и эксперементально найду математическое описание тембра. После чего попробую искусственно повлиять на тембр и синтезировать новое звучание.

Загружаю аудиофрагмент звукоизвлечения басовой ноты PE и вывожу на графиках левый и правый канал

```
clear all
close all
[x, fs] = audioread('Sound2.m4a');
sound(x, fs);
```

Что бы получить переменную соответствующую времени делаю следующее. Нахожу временной интервал между отчетами (dt). Через время между отчетами и кол-вом отчетов получаю время длительности сигнала (Ts). Создаю переменную времени (t)

```
dt = 1/fs;
N  = length(x);
Ts = (N-1)*dt;
t  = 0:dt:Ts;

figure
subplot(2,1,1);
plot(t, x(:,1), 'LineWidth', 2);
ylim([-0.75 0.75]);
xlim([0 t(end)]);
title('Right chanel of sound');
xlabel('t, s');

subplot(2,1,2);
plot(t, x(:,2), 'LineWidth', 2);
ylim([-0.75 0.75]);
xlim([0 t(end)]);
title('Left chanel of sound');
xlabel('t, s');
```



Спектр звука

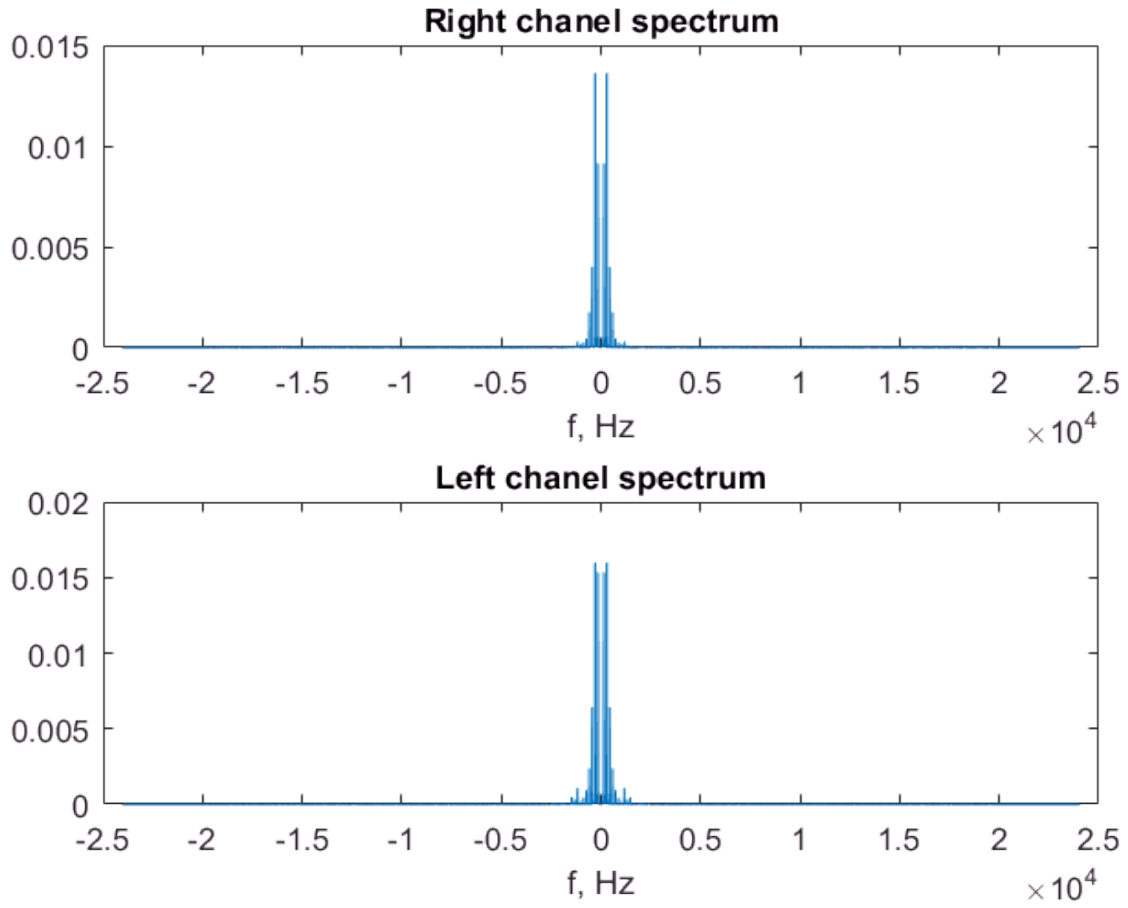
Для этого сначала введу переменную частоты. Она находится в пределах от -fs/2 до fs/2 и имеет столько же отчетов, сколько имеет входной сигнал (N)

```
df = fs/N;
Fm = fs/2;
f = -Fm:df:Fm - df;
```

Найду спектры каждого из каналов и обозначу их как Rs и Ls. Их удобно смещенные модули обозначу как Rsa и Lsa (right/left spectrum absolute)

```
Rs = fft(x(:,1));
Ls = fft(x(:,2));

Rsa = abs(fftshift(Rs)) / N;
Lsa = abs(fftshift(Ls)) / N;

figure
subplot(2,1,1);
plot(f, Rsa);
title('Right chanel spectrum');
xlabel('f, Hz');

subplot(2,1,2);
plot(f, Lsa);
title('Left chanel spectrum');
```

```
xlabel('f, Hz');
```



Отобразим эти спектры в масштабе.

Обозначим пороговую частоту, до которой нас интересует спектр (Fi - interest frequency) равной 2кГц

```
Fi = 2e3;
```

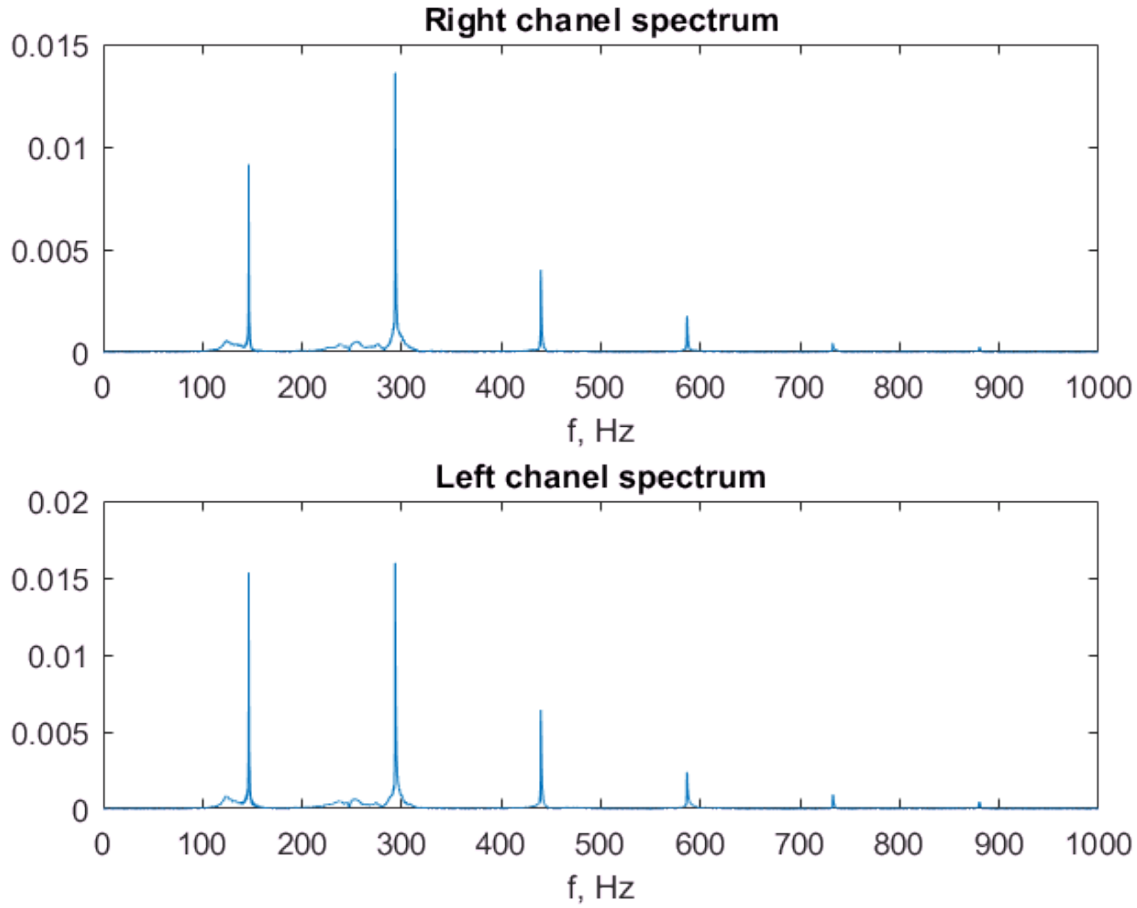
Нужно найти индекс переменной частоты, которым будет ограничен график (Ni). Зависимость индекса переменной частоты (f) от требуемой частоты (Fi) линейна и может быть выражена следующим выражением для положительных частот

```
Ni = (N/2) * (1 + Fi/fs);
```

Для показа положительных частот на графике введу переменную частоты отображающую только положительные частоты ограничиваясь интересующей частотой (Fi).

```
fp   = f  (N/2 + 1: Ni);
Rsap = Rsa(N/2 + 1: Ni);
Lsap = Lsa(N/2 + 1: Ni);

figure
subplot(2,1,1);
plot(fp, Rsap);
title('Right chanel spectrum');
xlabel('f, Hz');
```

```
subplot(2,1,2);
plot(fp, Lsap);
title('Left chanel spectrum');
xlabel('f, Hz');
```



Как видно из графиков спектр ноты PE наполнен кратными по частоте гармониками основного тона и характеристика их амплитуд не монотонно затухающая.

Некоторые векторы достаточно длинные, удалю ненужные для сбережения памяти

```
clear Rs Ls Rsa Lsa f;
```

Найду частоты каждой гармоники.

Для этого воспользуюсь функцией поиска пиков и задам минимальное расстояние между пиками обеспечивающее игнорирование шума и прочего.

Вектор Fharm будет заполнен частотами гармоник, для перевода частот в индексы используется вектор Nharm. В вектор pks будут записаны высоты каждой гармоники.

```
[pks, Fharm] = findpeaks(Rsap, fp, 'MinPeakDistance', 100, 'MinPeakHeight', 5e-5);
Nip   = Ni - N/2;
Nharm = floor(Nip * Fharm/Fi);
```

Вывожу спектр правого канала

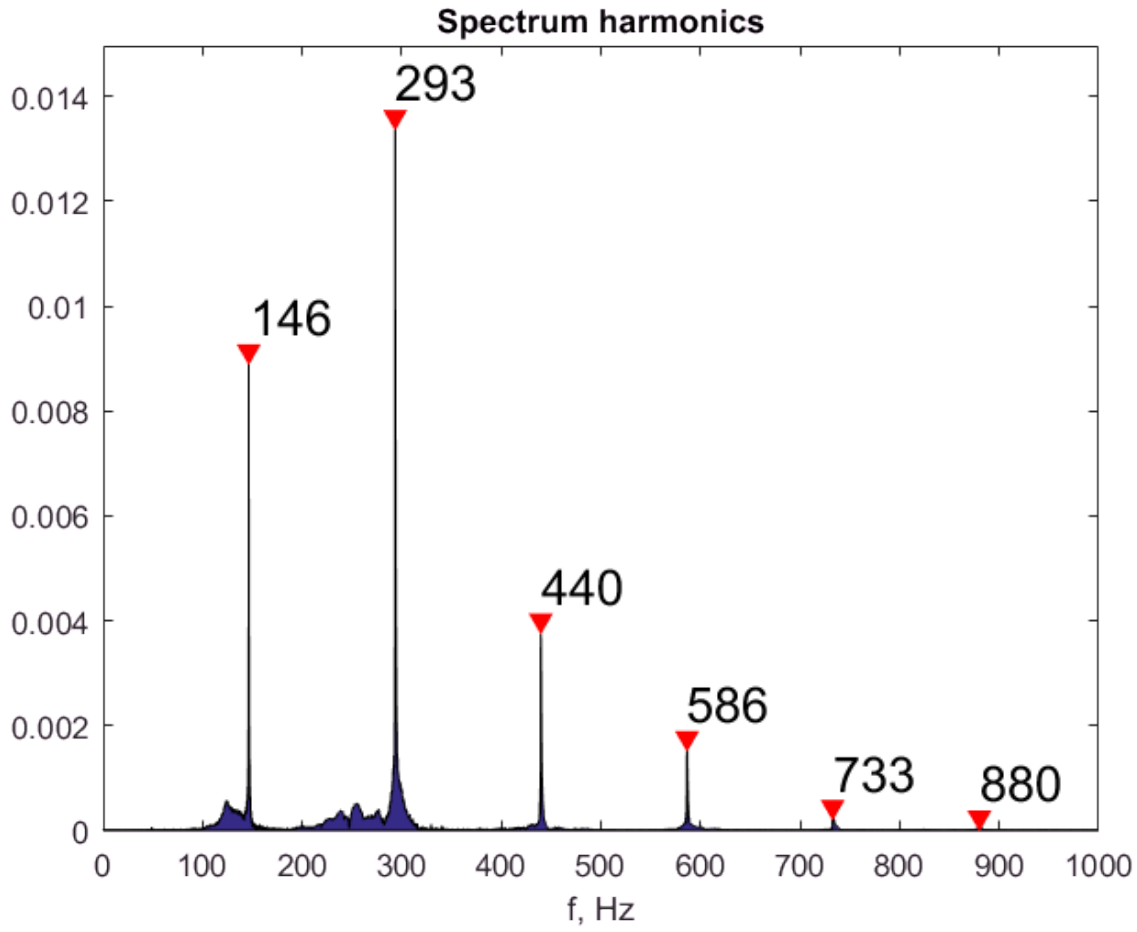
```
figure
area(fp, Rsap);
```

Поверх него накладываю фигуры, обозначающие положения пиков

```
hold on
plot(Fharm, pks, 'rv', 'MarkerFaceColor', 'r');
```

Создаю текст с частотами и накладываю текст на каждый пик спектра. Текст должен быть незначительно выше пиков для повышения читабельности, для чего был введен параметр масштаба yScaleAdd, поднимающий надписи на 5%

```
yScaleAdd = max(pks)*0.05;

cellpeaks = cellstr(num2str(round(Fharm', 0)));
text(Fharm, yScaleAdd+pks, cellpeaks, 'FontSize', 16);
ylim([0 max(pks)+2*yScaleAdd]);
hold off
title('Spectrum harmonics');
xlabel('f, Hz');
```



Воспользовавшись таблицей соответствия нотам частот можно определить слышимые ноты

Ноты	Суббконтр-октава	Контр-октава	Большая	Малая	Первая	Вторая	Третья	Четвертая	Пятая
ДО	16,35	32,70	65,41	130,82	261,63	523,26	1046,52	2093,04	4186,08
ДО диез	17,32	34,65	69,30	138,59	277,18	554,36	1108,72	2217,44	4434,88
РЕ	18,35	36,71	73,42	146,83	293,66	587,32	1174,64	2349,28	4698,56
РЕ диез	19,45	38,89	77,78	155,57	311,13	622,26	1244,52	2489,04	4978,08
МИ	20,60	41,20	82,41	164,82	329,63	659,26	1318,52	2637,04	5274,08
ФА	21,83	43,65	87,31	174,62	349,23	698,46	1396,92	2793,84	5587,68
ФА диез	23,12	46,25	92,50	185,00	369,99	739,98	1479,96	2959,92	5919,84
СОЛЬ	24,50	49,00	98,00	196,00	392,00	784,00	1568,00	3136,00	6272,00
СОЛЬ диез	25,96	51,91	103,83	207,65	415,30	830,60	1661,20	3322,40	6644,80
ЛЯ	27,50	55,00	110,00	220,00	440,00	880,00	1760,00	3520,00	7040,00
ЛЯ диез	29,14	58,27	116,54	233,08	466,16	932,32	1864,64	3729,28	7458,56
СИ	30,87	61,74	123,47	246,94	493,88	987,76	1975,52	3951,04	7902,08

Найду отношение гармоник к основному тону

Из пиков видно, что основной тон соответсвует первой гармонике, основной тон обозначу как F0.
(Fr - frequency ratio)

```
F0 = Fharm(1);
Fr = Fharm ./ F0;

strMessage = ['Harmonics frequency ratio ', num2str(Fr)];
disp(strMessage);
```

```
Harmonics frequency ratio 1      2.0037      3.0025      4.0049      5.0037      6.0099
```

С точностью до тысячных частоты гармоник кратные основной.

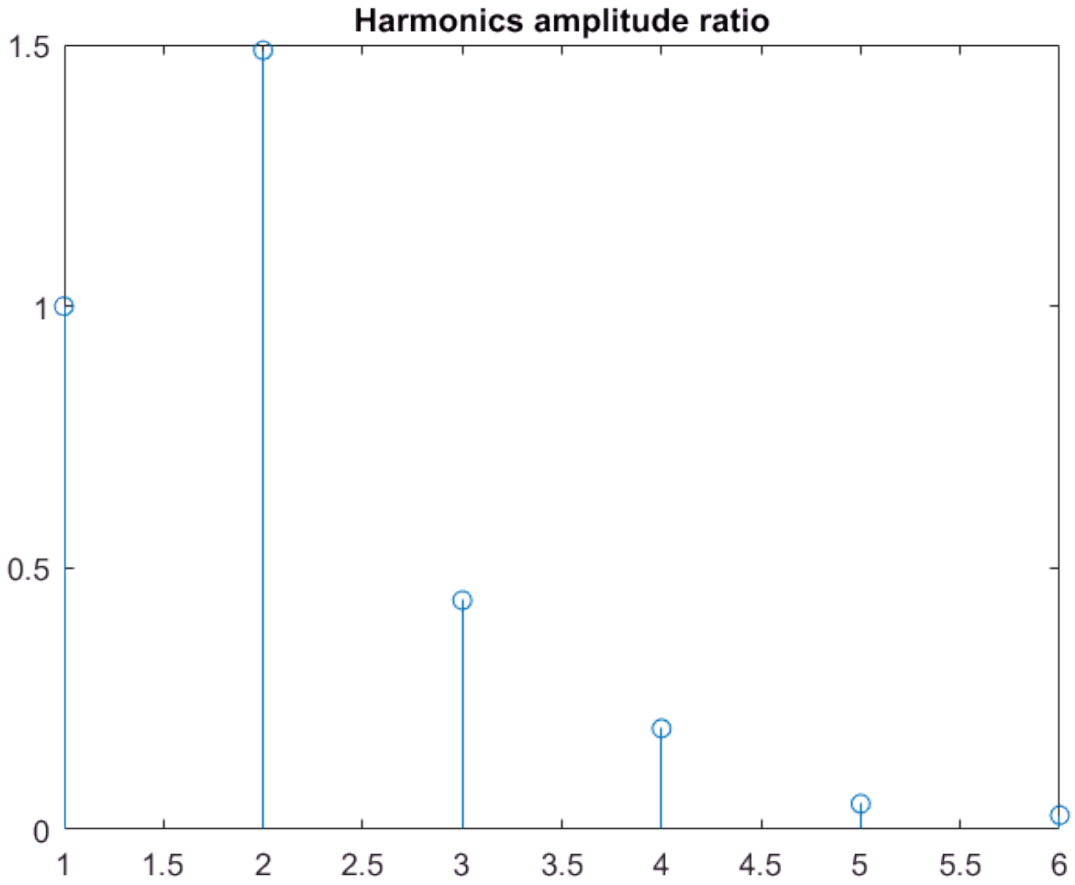
Найду отношения амплитуд гармоник к амплитуде основной.

(Ar - amplitude ratio)

```
Ar = (pks ./ pks(1) )';
strMessage = ['Harmonics amplitude ratio ', num2str(Ar)];
disp(strMessage);
```

```
Harmonics amplitude ratio 1      1.4889     0.43814     0.19305    0.049476    0.027427
```

```
figure
stem(Ar);
title('Harmonics amplitude ratio')
```



1.1 Шумоподавление и Децимация

В звукозаписи есть записанная часть, где струна гитары не звучит. Используя этот участок звука можно получить постоянный шум внешней среды, который сопровождает звукоряд от начала и до конца. Идея уменьшения постоянного шума - найти спектр шума, из него найти уровень спектра шумов. Далее из исследуемых спектров можно будет вычесть уровень шумов, чем будет повышен уровень точности подсчетов.

Децимация, или уменьшение частоты дискретизации, может быть использовано для меньшего охвата спектра белого шума, а так же для более точного анализа при оконном преобразовании фурье. Заранее известно, что более 6той гармоники (880 Гц) извлечь вряд ли получится, а значит можно ограничиться частотой дискретизации 3кГц, взятой с запасом

Децимация

```
clear all
close all
[x, fs] = audioread('Sound2.m4a');

dt = 1/fs;
N  = length(x);
Ts = (N-1)*dt;
t  = 0:dt:Ts;

df = fs/N;
```

```
Fm = fs/2;
f = -Fm:df:Fm - df;
Fc = 3000;

Rs = fft(x(:,1));
Ls = fft(x(:,2));
```

Создаю идеальный ФНЧ

```
H = zeros(1,N);
```

Для положительных частот индекс соответствующей частоты среза в векторе частот находится как

```
Nc = ceil((N/(2*Fm))*Fc + N/2);
disp(['Cutoff frequency - ', num2str(f(Nc))])
```

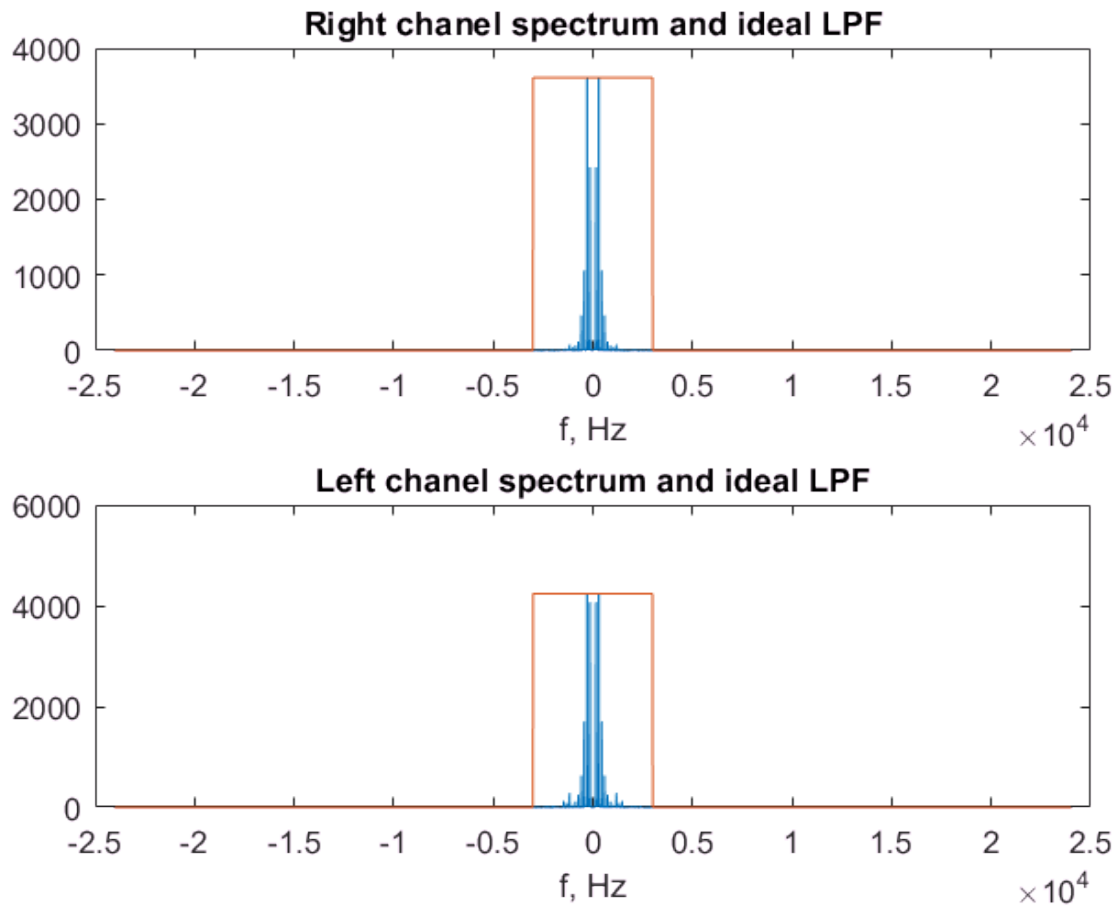
```
Cutoff frequency - 2999.819
```

Начальный и конечный индекс пропускания фильтра

```
Ns = N/2 - (Nc - N/2);
Ne = N/2 + (Nc - N/2);

H(Ns:Ne) = ones(1,Ne-Ns+1);
```

Изображаю спектры правого и левого канала звукоряда с наложенным идеальным ФНЧ

```
figure
subplot(2,1,1);
plot(f, abs(fftshift(Rs)));
hold on
plot(f, H * max(abs(Rs)));
title('Right chanel spectrum and ideal LPF');
xlabel('f, Hz');

subplot(2,1,2);
plot(f, abs(fftshift(Ls)));
hold on
plot(f, H * max(abs(Ls)));
title('Left chanel spectrum and ideal LPF');
xlabel('f, Hz');
```



Перемножаю спектры с идеальным фильтром и возвражаю спектры во временную область для оценки искажений

```
Rf = fftshift( fftshift(Rs) .* H' );
Lf = fftshift( fftshift(Ls) .* H' );
```

Из-за подсчетов результат окажется комплексным, но его мнимая часть будет на несколько порядков меншьа реальной части, так что ей можно пренебречь.

```
xf(:,1) = real(ifft(Rf));
xf(:,2) = real(ifft(Lf));
```

Проводим децимацию

```
decRate = ceil(Fm/Fc);
fsd = fs /decRate;
xd(:,1) = decimate(xf(:,1), decRate, 'fir');
xd(:,2) = decimate(xf(:,2), decRate, 'fir');
disp(['Sampling freqency redused by - ', num2str(decRate)])
```

```
Sampling freqency redused by - 8
```

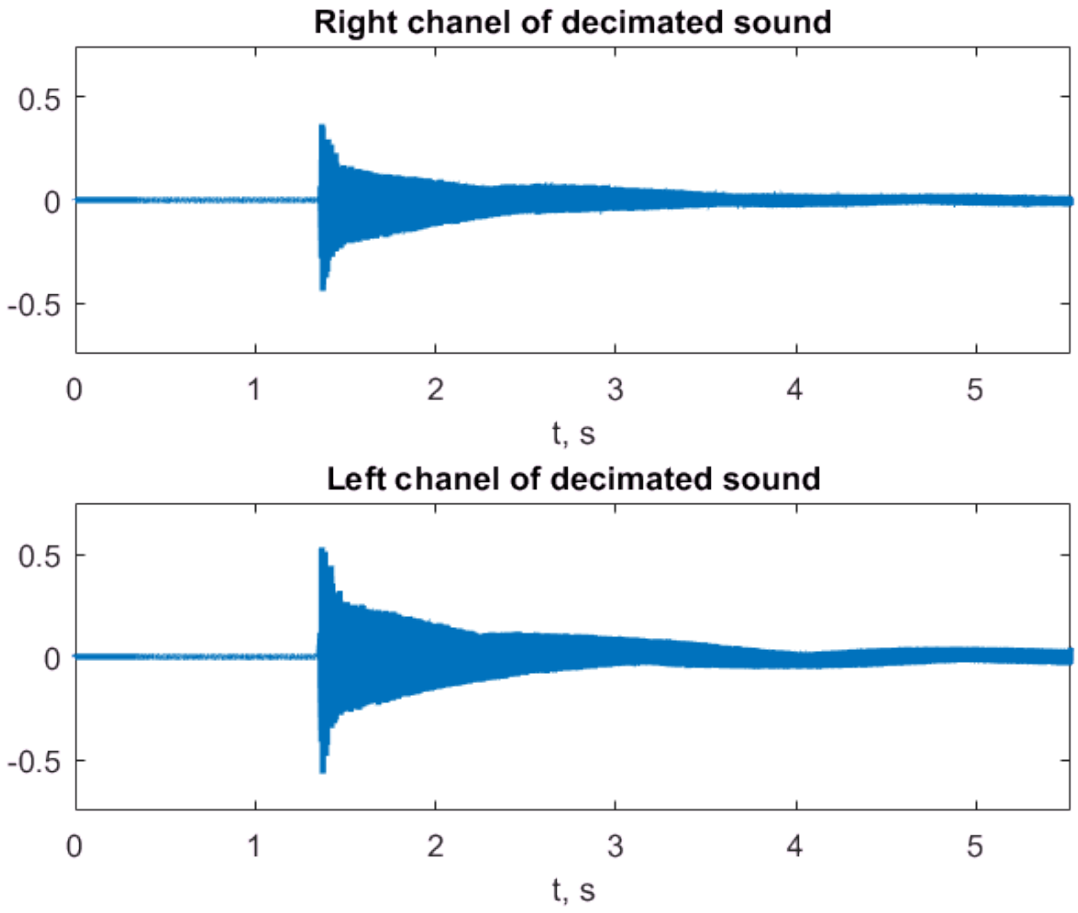
После децимации звук практически не потерял качество

```
sound(xd, fsd);
```

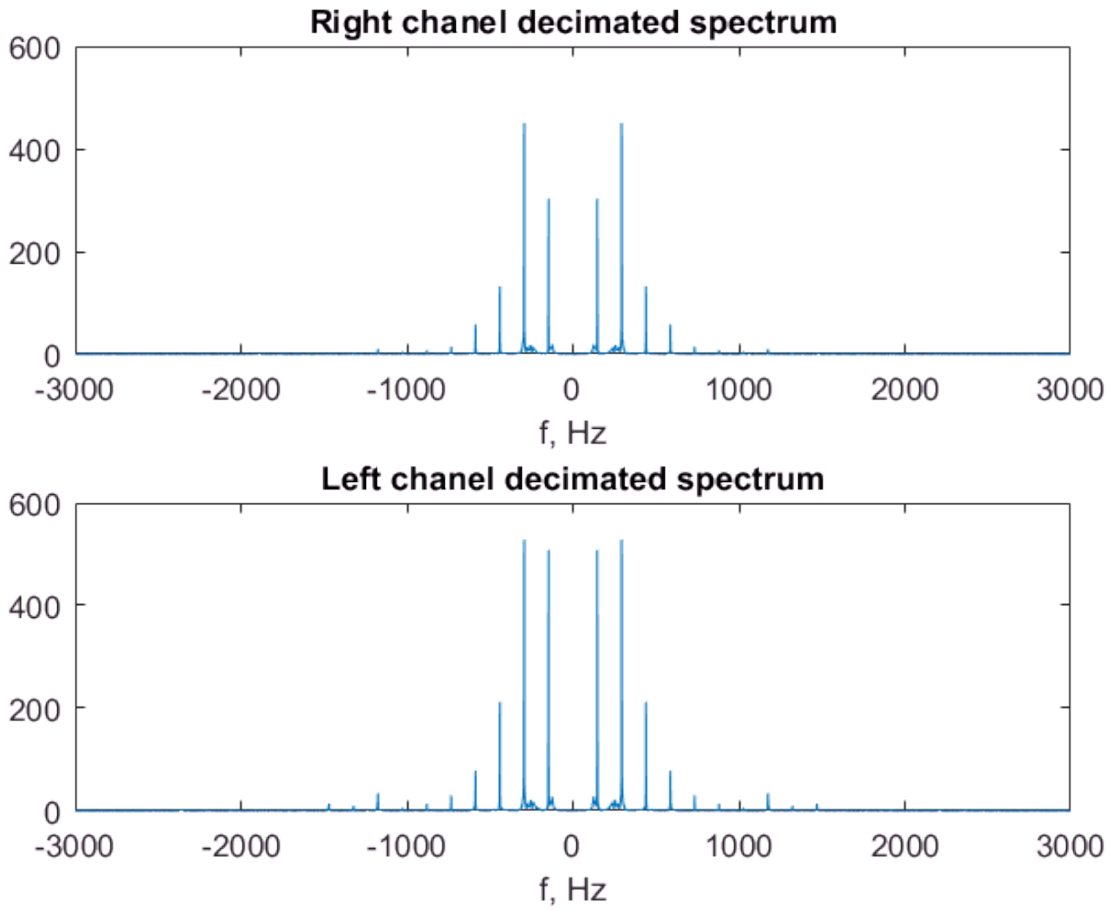
```
sound(x, fs);

clear H xf Nc Ne Ns Rf Lf Rs Ls f df;
```

Спектр децемируемого звукоряда и временное представление

```
Rd = fft(xd(:,1));
Ld = fft(xd(:,2));

N  = N / decRate;
df = fsd/N;
f  = -Fc:df:Fc-df;
dt = 1/fsd;
t  = 0:dt:Ts;

figure
subplot(2,1,1);
plot(t, xd(:,1), 'LineWidth', 2);
ylim([-0.75 0.75]);
xlim([0 t(end)]);
title('Right chanel of decimated sound');
xlabel('t, s');

subplot(2,1,2);
plot(t, xd(:,2), 'LineWidth', 2);
ylim([-0.75 0.75]);
xlim([0 t(end)]);
title('Left chanel of decimated sound');
xlabel('t, s');
```




```
figure
subplot(2,1,1);
plot(f, abs(fftshift(Rd)));
title('Right chanel decimated spectrum');
xlabel('f, Hz');

subplot(2,1,2);
plot(f, abs(fftshift(Ld)));
title('Left chanel decimated spectrum');
xlabel('f, Hz');
```

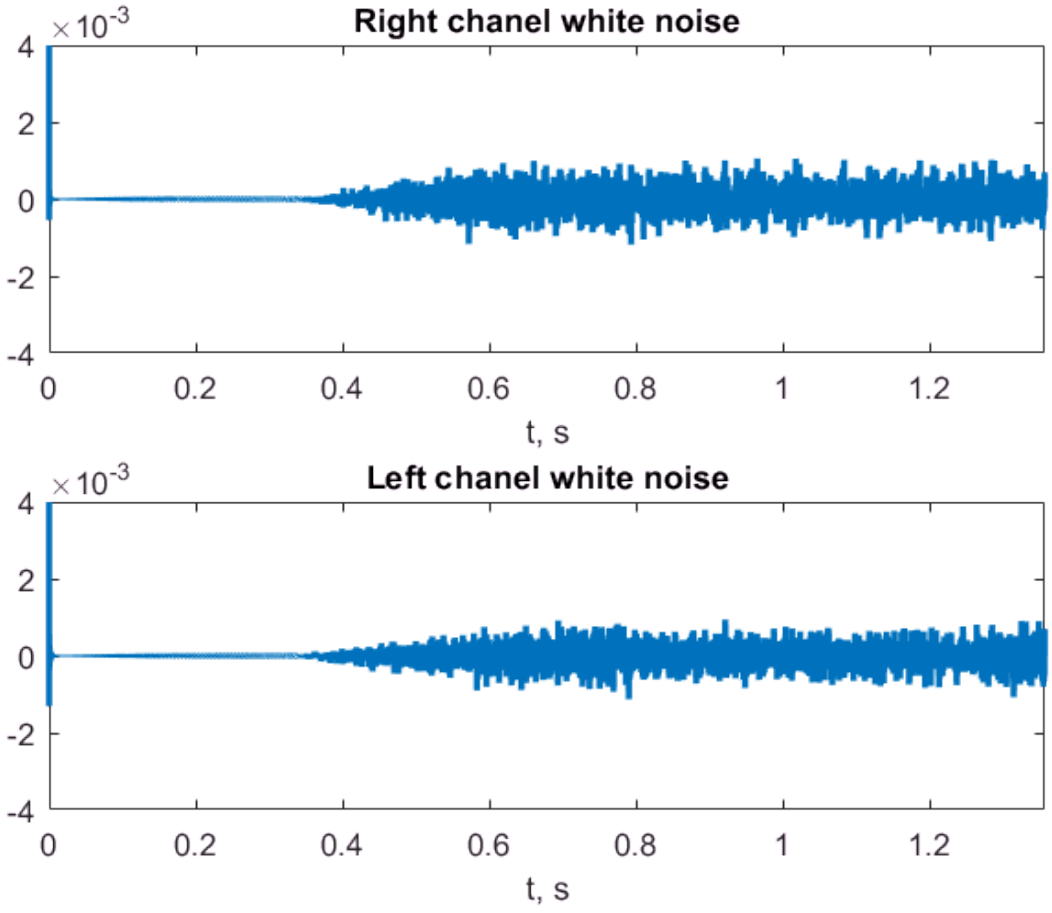


Шумоподавление ЗАКЛАДКА 1

Эмпирическим путем установлено, что амплитуда шума не поднимается более 0.04, так мы можем найти длинну чистого шума и получить его спектр

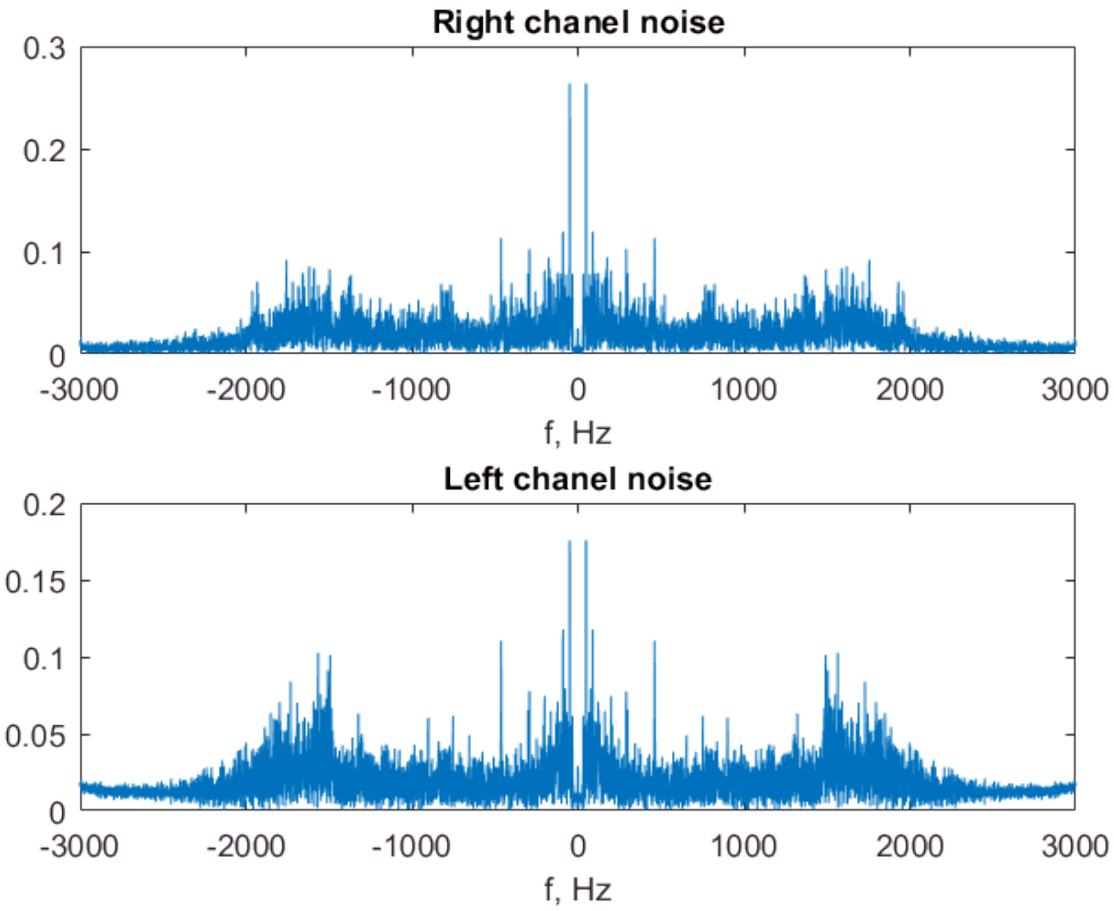
```
Nn        = 1;
threshold = 0.04;
for i = 1:N
    if xd(i) > threshold
        Nn = i - 40;
        break;
    end
end

figure
subplot(2,1,1);
plot(t(1:Nn), xd(1:Nn,1), 'LineWidth', 2);
ylim([-threshold threshold]/10);
xlim([0 t(Nn)]);
title('Right chanel white noise');
xlabel('t, s');

subplot(2,1,2);
plot(t(1:Nn), xd(1:Nn,2), 'LineWidth', 2);
ylim([-threshold threshold]/10);
xlim([0 t(Nn)]);
title('Left chanel white noise');
```

```
xlabel('t, s');
```



Спектр шума

```
Nr = fft(xd(1:Nn,1));
Nl = fft(xd(1:Nn,2));

Fc  = 3000;
dfn = 2*Fc/Nn;
fn  = -Fc:dfn:Fc - dfn;

figure
subplot(2,1,1);
plot(fn, abs(fftshift(Nr)));
title('Right chanel noise');
xlabel('f, Hz');

subplot(2,1,2);
plot(fn, abs(fftshift(Nl)));
title('Left chanel noise');
xlabel('f, Hz');
```



Если предположить, что спектр белого шума будет примерно постоянным на протяжении всего звукоряда - то можно растянуть спектр шума аппроксимацией или интерполяцией и отнять из общего спектра

```
[Nrpol, Sr, mur] = polyfit(fn, Nr', 80);
```

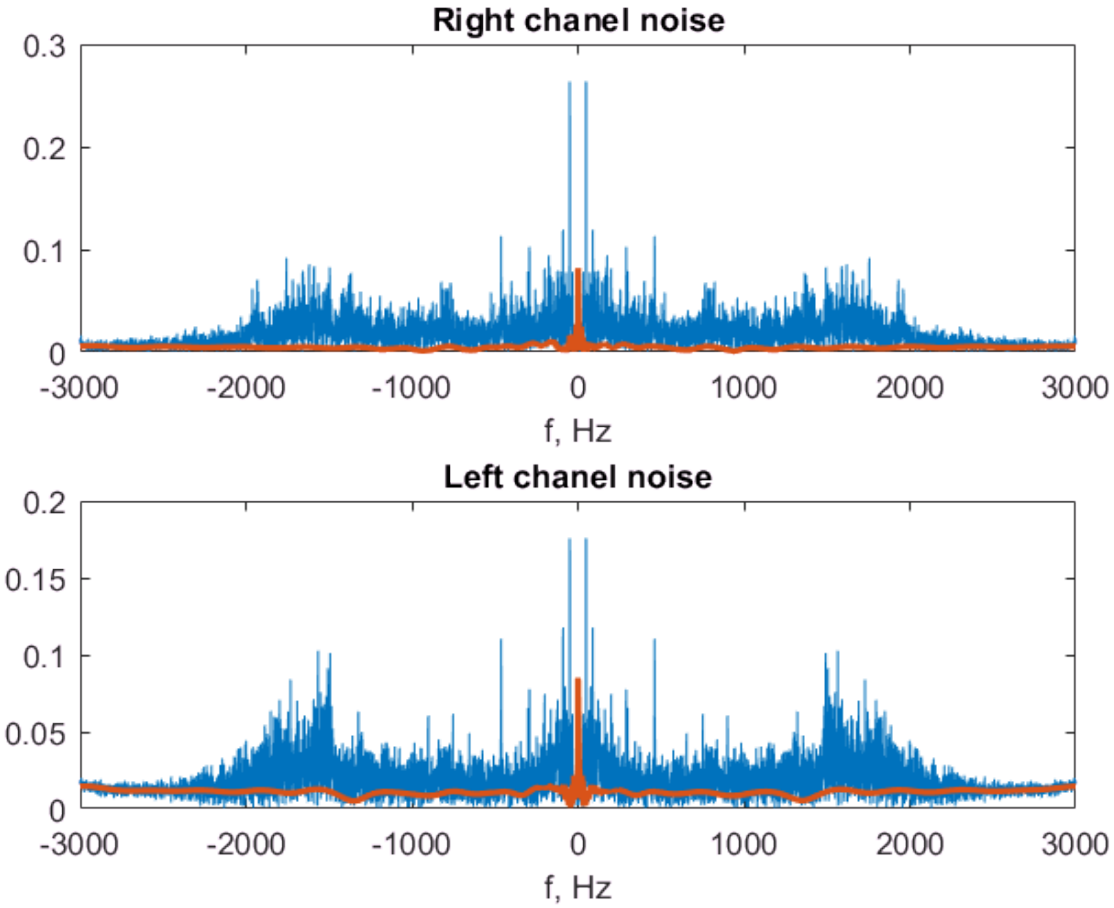
```
Warning: Polynomial is badly conditioned. Add points with distinct X values or reduce the
degree of the polynomial.
```

```
[Nlpol, Sl, mul] = polyfit(fn, Nl', 80);
```

```
Warning: Polynomial is badly conditioned. Add points with distinct X values or reduce the
degree of the polynomial.
```

```
Nra = polyval( Nrpol, f, Sr, mur );
Nla = polyval( Nlpol, f, Sl, mul );

figure
subplot(2,1,1);
plot(fn, abs(fftshift(Nr)));
hold on
plot(f, abs(fftshift(Nra)), 'LineWidth', 2);
title('Right chanel noise');
xlabel('f, Hz');

subplot(2,1,2);
```

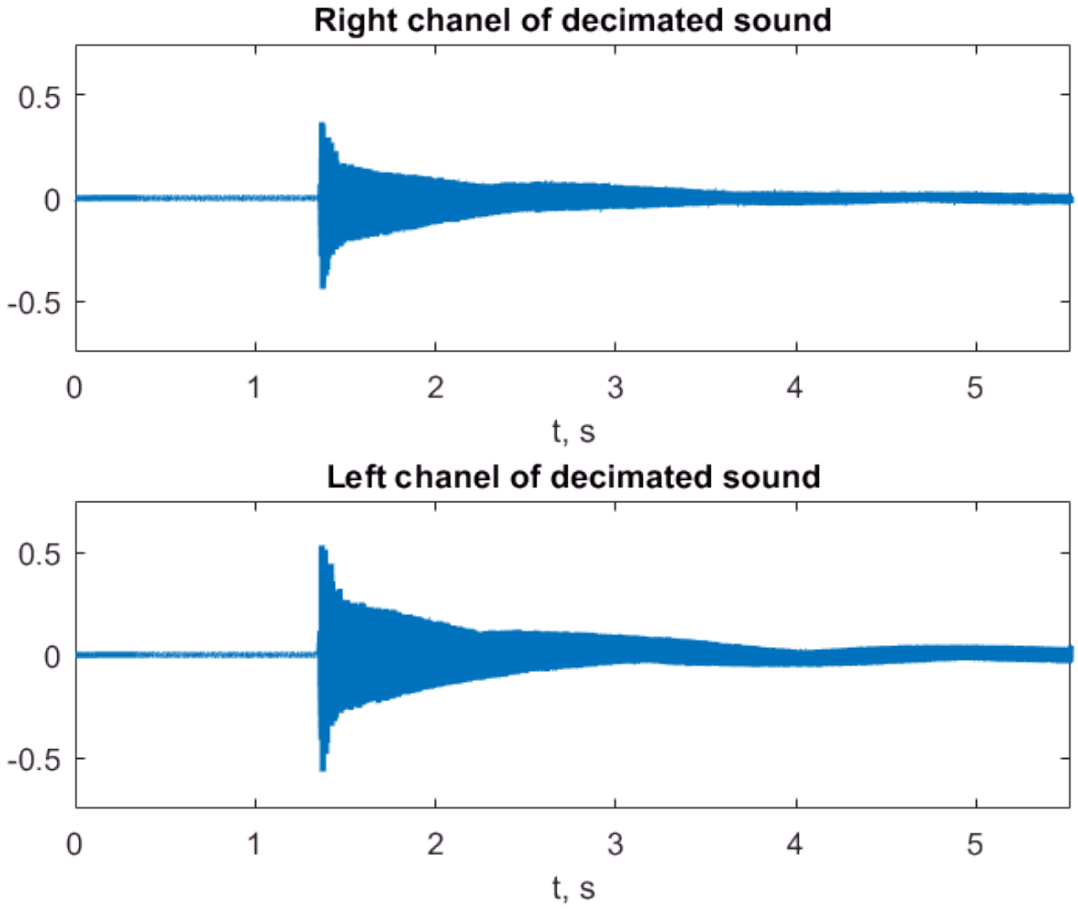
```
plot(fn, abs(fftshift(Nl)));
hold on
plot(f, abs(fftshift(Nla)), 'LineWidth', 2);
title('Left chanel noise');
xlabel('f, Hz');
```



А теперь отнимаю из общего спектра спектр шума и смотрим что получится

```
Xd(:,1) = fft(xd(:,1));
Xd(:,2) = fft(xd(:,2));

Xd(:,1) = Xd(:,1) - Nra';
Xd(:,2) = Xd(:,2) - Nla';

xdf(:,1) = real(ifft(Xd(:,1)));
xdf(:,2) = real(ifft(Xd(:,2)));

sound(xdf, fs);

figure
subplot(2,1,1);
plot(t, xdf(:,1), 'LineWidth', 2);
ylim([-0.75 0.75]);
xlim([0 t(end)]);
title('Right chanel of decimated sound');
xlabel('t, s');

subplot(2,1,2);
```

```
plot(t, xdf(:,2), 'LineWidth', 2);
ylim([-0.75 0.75]);
xlim([0 t(end)]);
title('Left chanel of decimated sound');
xlabel('t, s');
```

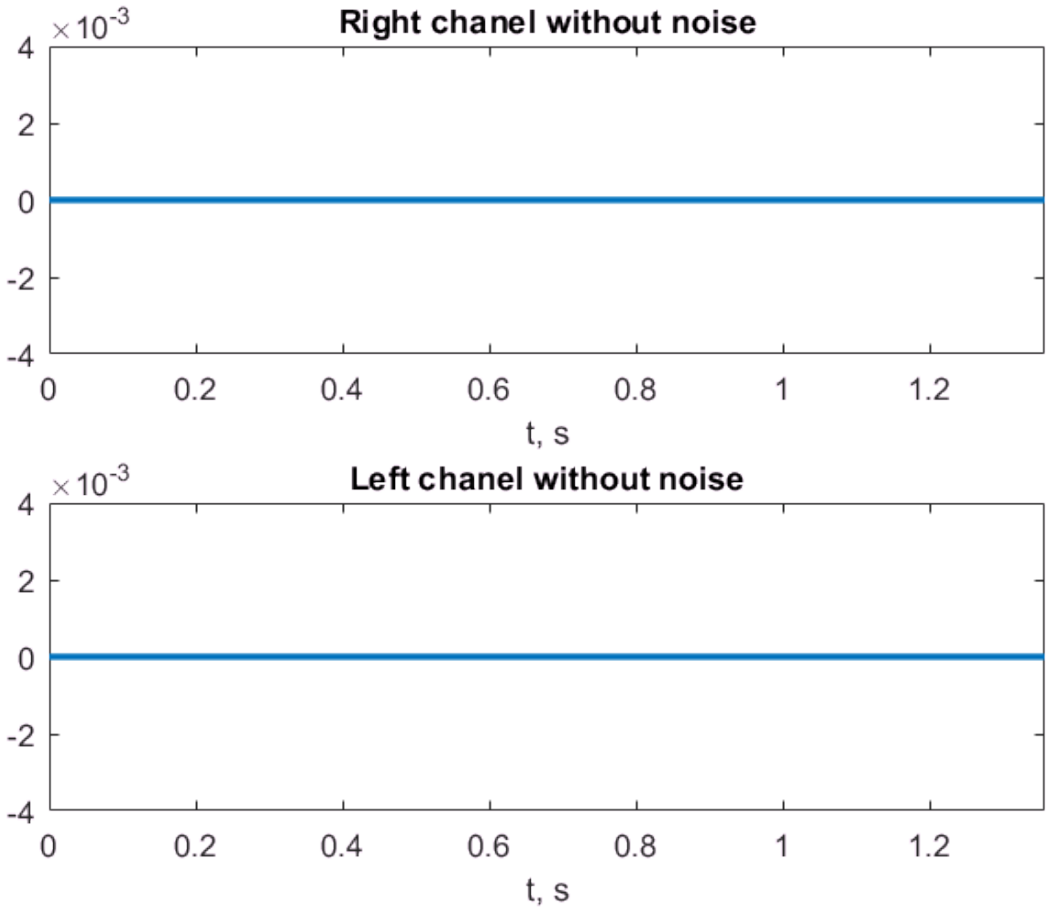


Шумоподавление ЗАКЛАДКА 2

Создаю спектр с противофазным шумом

```
xn(:,1) = - ifft(Nr);
xn(:,2) = - ifft(Nl);

xd(1:Nn,1) = xd(1:Nn,1) + xn(:,1);
xd(1:Nn,2) = xd(1:Nn,2) + xn(:,2);

figure
subplot(2,1,1);
plot(t(1:Nn), xd(1:Nn,1), 'LineWidth', 2);
ylim([-threshold threshold]/10);
xlim([0 t(Nn)]);
title('Right chanel without noise');
xlabel('t, s');

subplot(2,1,2);
plot(t(1:Nn), xd(1:Nn,2), 'LineWidth', 2);
ylim([-threshold threshold]/10);
```

```
xlim([0 t(Nn)]);
title('Left chanel without noise');
xlabel('t, s');
```



2. Временная зависимость амплитуд гармоник

Как известно музыкальные звуки одних и тех же нот отличаются друг от друга на слух раличным тембром. Тембр описывается отношением гармоник и, как утверждают, во времени звучания тембр звука меняется и он меняется уникально для каждого инструмента. В этой части будет найдены временные зависимости тембра.

Такую зависимость можно найти используя оконное преобразование фурье и анализ гармоник, приведенный выше. Так можно получить амплитуды каждой гармоники в заданых временных промежутках и найти их временные изменения. Поскольку нельзя единовременно увеличивать уменьшать шаг частот преобразования фурье и уменьшать время наблюдения - прийдется выбрать оптимальное количество делений звукового сигнала так, что бы спектры временных промежутков несли полезную информациб и качество этой информации было высоким.

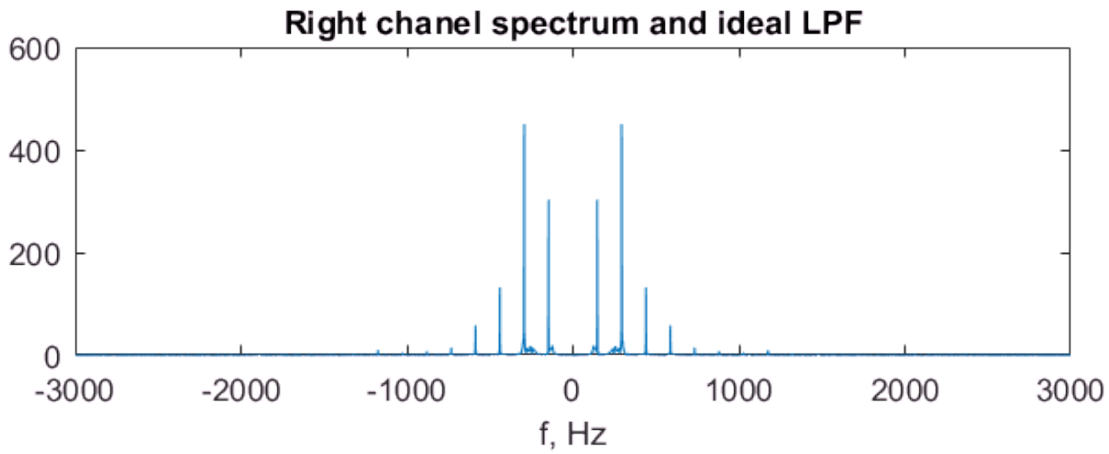
Утверждают, что большая часть информации находится в временном промежутке аттаки звука. В этот период темб меняется быстрее всего и несет в себе много информации. А значит длительность аттаки может служить критерием оптимального количества разбитых временных промежутков.

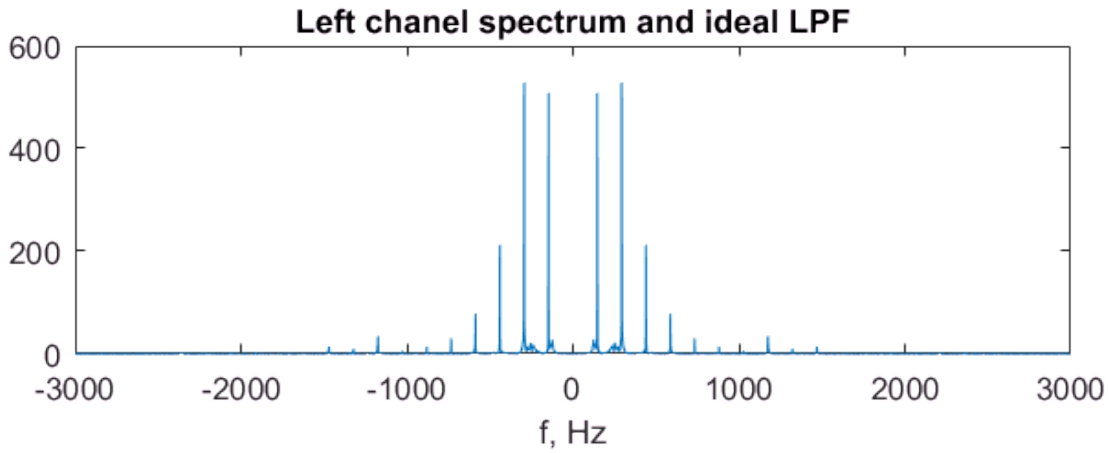
Нахожу длительность аттаки

```
clear all
close all
clc
[x, fs] = audioread('Sound2.m4a');
```

Все дальнейшие расчеты будут проводиться над децимированым цифровым сигналом. В функцию decimatef заложен код, описаный в пункте выше.

```
decRate = 8;
xd      = decimatef( x, fs, decRate );
```



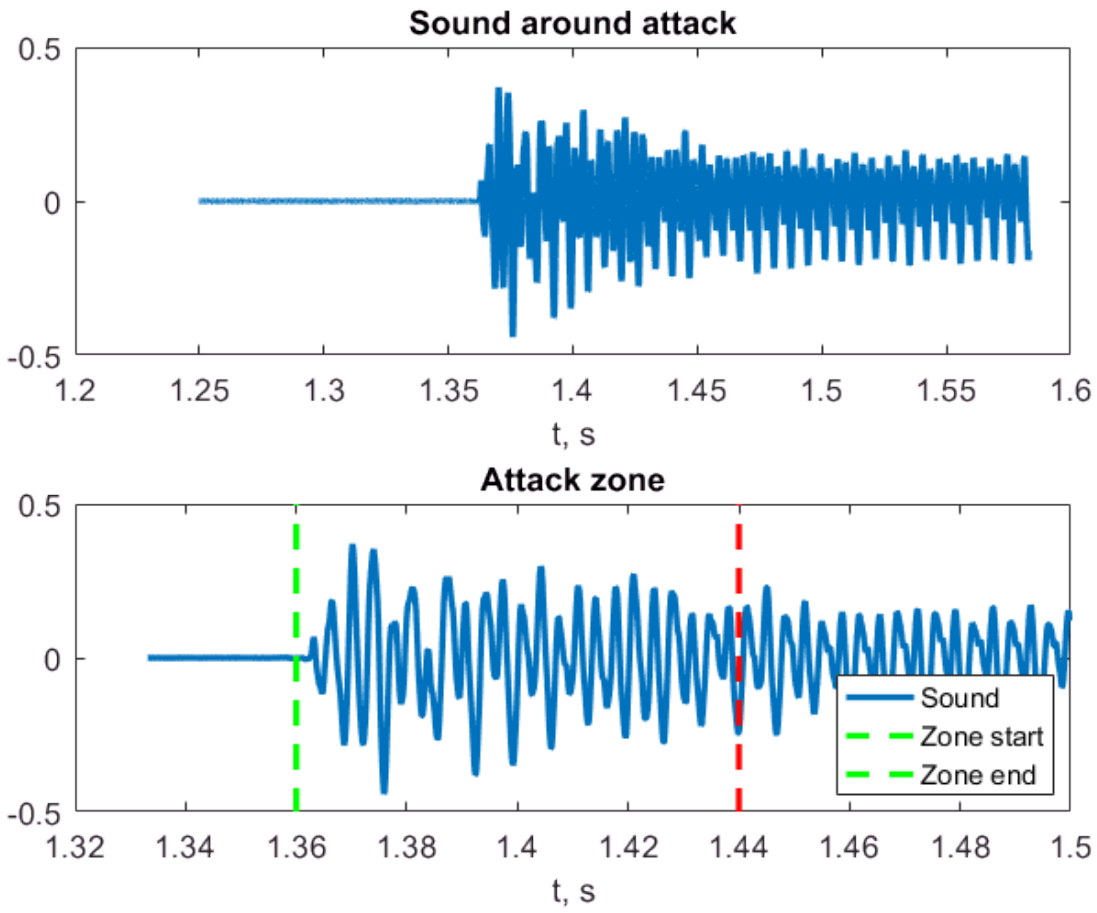



```
clear x;

fs = fs/decRate;
dt = 1/fs;
N  = length(xd(:,1));
Ts = (N-1)*dt;
t  = 0:dt:Ts;
```

*Точки взяты графически

```
StartAttack = ones(100)*1.36;
EndAttack   = ones(100)*1.44;
Amplitude   = linspace(-0.5, 0.5, 100);

figure
subplot(2,1,1);
```

```
plot(t(75e2:1:95e2), xd(75e2:1:95e2, 1), 'LineWidth', 2);
title('Sound around attack');
xlabel('t, s');

subplot(2,1,2);
plot(t(80e2:1:90e2), xd(80e2:1:90e2,1), 'LineWidth', 2);
hold on
plot(StartAttack, Amplitude, '--g', 'LineWidth', 2);
plot(EndAttack, Amplitude, '--r', 'LineWidth', 2);
hold off
title('Attack zone');
xlabel('t, s');
legend('Sound', 'Zone start', 'Zone end', 'Location', 'Best');
```



Экспериментально была найдено, что время аттаки длится примерно 80мс (1360 - 1440). Этот временной промежуток обозначен вертикальными линиями на графике. Для получения полезной информации было выбрано разбить участок аттаки минимум на 6 частей (длительностью dT). Найду количество отчетов, попадающих в один временной промежуток (Nw - number of samples in window). Ta - время атаки. Na - кол-во окон в атаке. Nfrag - кол-во окон/фрагментов для всего сигнала. Nw - кол-во отчетов для одного окна

```
Ta = 0.080;
Na = 6;
dT = Ta / Na; % Time revolution

Nw    = floor((dT/Ts)*N + 1);
Nfrag = floor(N/Nw);
```

Теперь создаю временную матрицу, где в каждой строке будет Nw отчетов времени, а количеству строк будет соответствовать количесвто фрагментов, на которые разбит аудиофайл

```
T = reshape(t(1:Nfrag*Nw), Nfrag, Nw);
```

Оцениваю разрешение частот

Произведение шага частот на длительность одного временного фрагмента равна 1 по определению. Из чего нахожу шаг частотной сетки.

```
df = fs / Nw; % Frequency resolution
```

Проверка определения (произведение должно быть равным единице)

```
strMessage = ['df * dT = ', num2str(df * dT)];
disp(strMessage);
```

```
df * dT = 0.98765
```

```
strMessage = ['Frequency step is ', num2str(df)];
disp(strMessage);
```

```
Frequency step is 74.0741
```

Если шаг частот будет больше основной гармоники (146 Hz), то могут возникнуть проблемы извлечения гармоник из спектра. Следует сделать проверку

```
F0 = 146
```

```
F0 = 146
```

```
if df > F0/2
    warning('Frequency resolution to poor. It is bigger than half of base-tone. Try to make wi
end
```

```
Warning: Frequency resolution to poor. It is bigger than half of base-tone. Try to make
window wider
```

Матрица спектров

Для каждого временного промежутка надо найти спектр и записать его в соответствующей строке матрицы Rs

```
Rs   = zeros(Nfrag, Nw);
Rsa  = zeros(Nfrag, Nw);
Rsap = zeros(Nfrag, floor(Nw/2));
for i = 1:Nfrag
    ns = (i-1)*Nw + 1;
    ne = i*Nw;
    Rs(i,:)               = fft(xd(ns:ne,1));
    Rsa(i,:)              = fftshift(Rs(i,:)) / N;
    Rsap(i,1:floor(Nw/2)) = Rsa(i,floor(Nw/2)+1:Nw-1);
end
```

```
fp  = 0:df:fs/2-df;
Fi  = 2e3;
Ni  = floor(Nw*(Fi/fs));
```

Тут можно просмотреть анимацию изменения спектра по времени

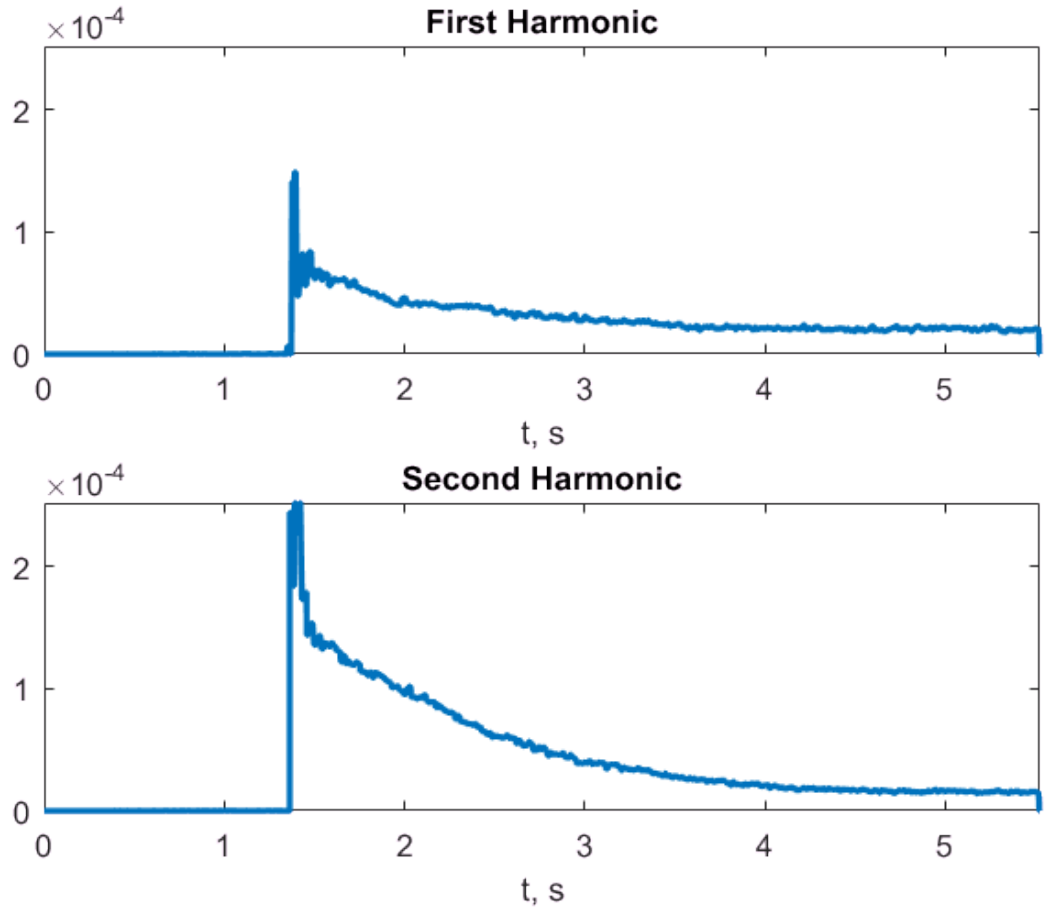
```
% v = VideoWriter('SpectrumInTimeNa.avi');
% open(v);
% figure
% for i = 1:Nfrag
%     area(fp(1:Ni), abs(Rsap(i,1:Ni)));
%     ylim([0 15e-5])
%     M(i) = getframe;
%     drawnow
% end
% writeVideo(v,M);
% movie(M, 1, 15);
% close(v);
```

Теперь создам матрицу временных зависимостей первых 6 гармоник от времени (т.к. они более выражены) Ввожу уже известные частоты этих гармоник как опорные (reference)

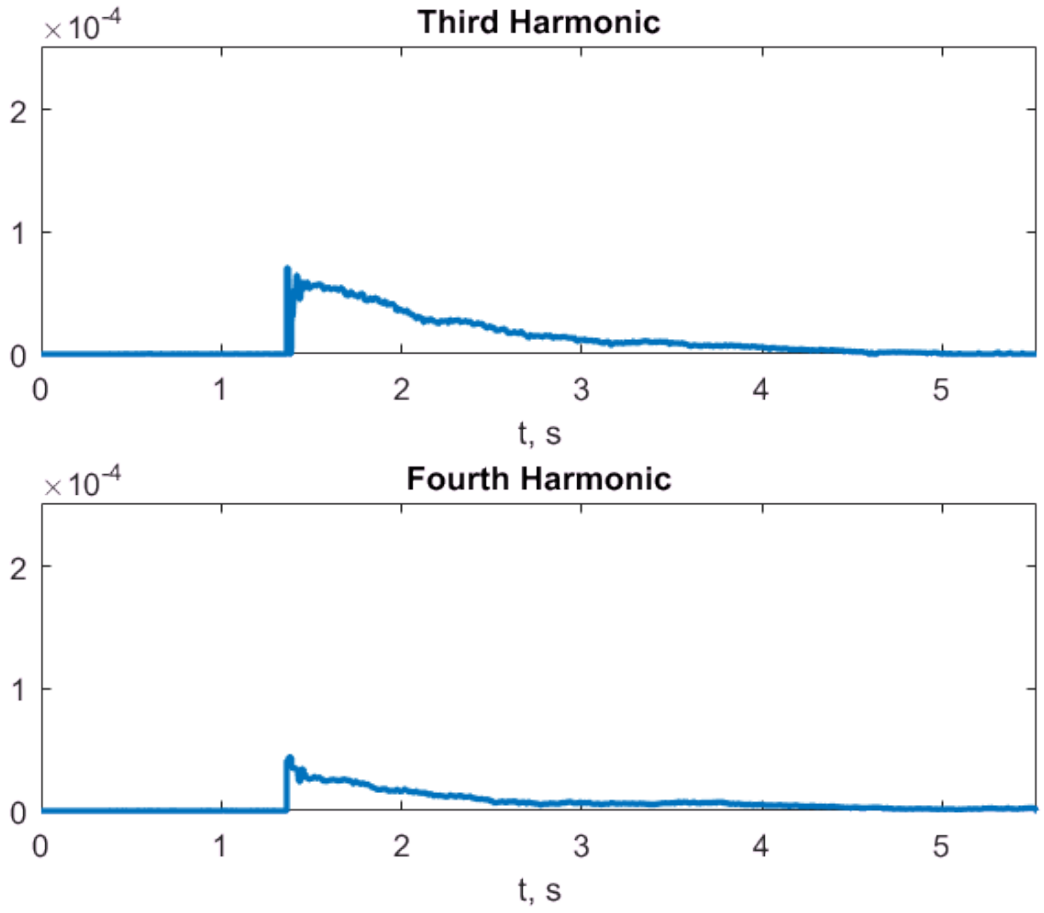
```
Harm = zeros(4, N);
F0   = 146;
Fref = F0 * [1:6];
```

Если выбранный пик соответствует одной из исследуемых гармоник - заполню временное окно (dT) гармоники значением пика (pks).

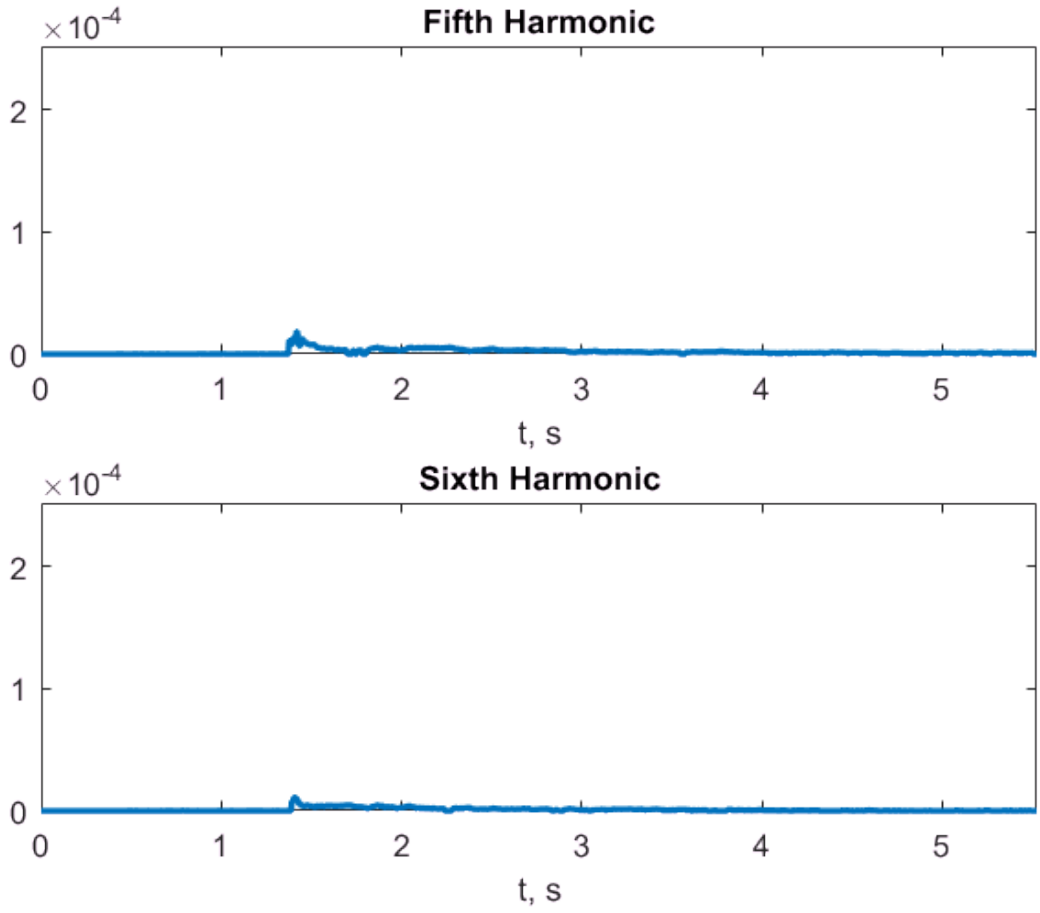
```
for i = 1:Nfrag
    [pks, Fharm] = findpeaks(abs(Rsap(i,:)), fp, 'MinPeakDistance', 0.8*F0);

    for j = 1:length(Fharm)
        for z = 1:length(Fref)
            if abs(Fharm(j) - Fref(z)) < 60
                ns = (i-1)*Nw + 1;
                ne = i*Nw;
                Harm(z, ns:ne) = ones(1,Nw) * pks(j);
            end
        end
    end
end
maxY = max(max(Harm)');

figure
subplot(2,1,1);
plot(t, Harm(1,:), 'LineWidth', 2);
ylim([0 maxY]);
xlim([0 t(end)]);
title('First Harmonic');
xlabel('t, s');

subplot(2,1,2);
plot(t, Harm(2,:), 'LineWidth', 2);
ylim([0 maxY]);
xlim([0 t(end)]);
title('Second Harmonic');
xlabel('t, s');
```



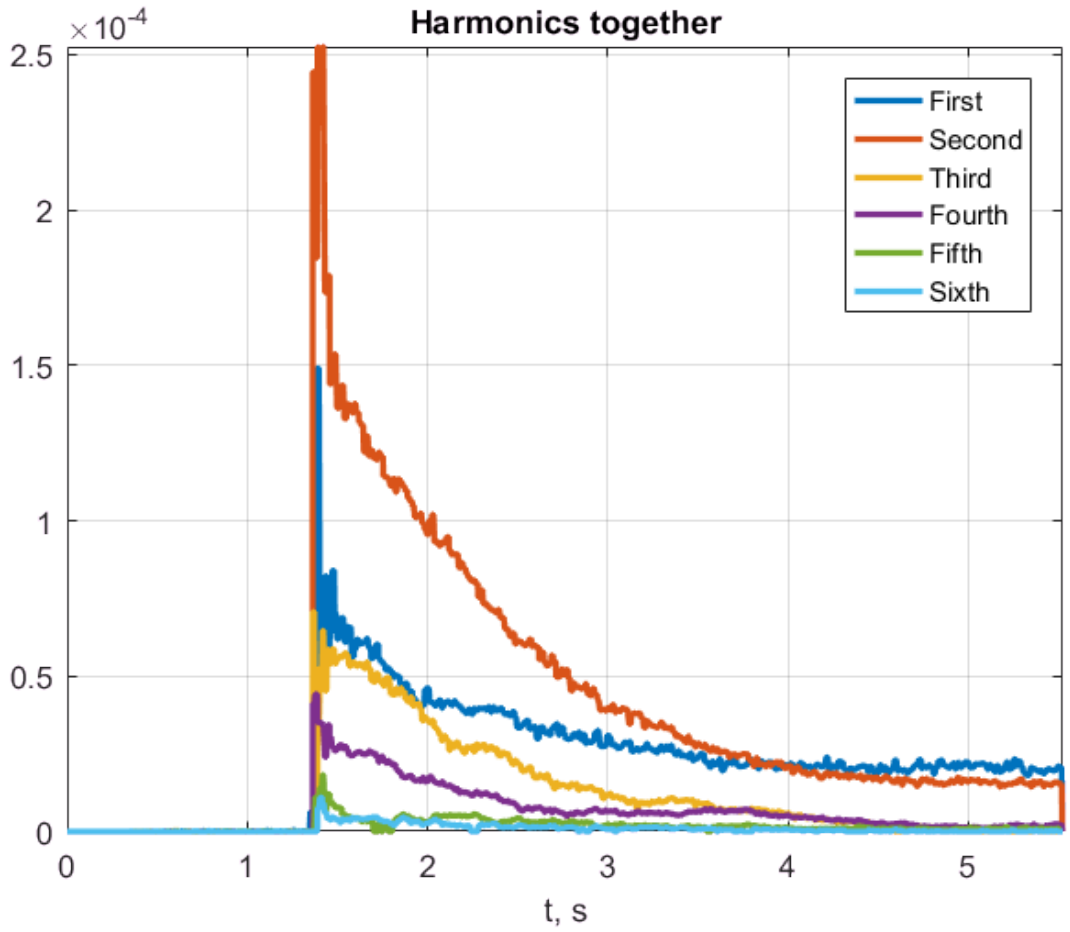

```
figure
subplot(2,1,1);
plot(t, Harm(3,:), 'LineWidth', 2);
ylim([0 maxY]);
xlim([0 t(end)]);
title('Third Harmonic');
xlabel('t, s');

subplot(2,1,2);
plot(t, Harm(4,:), 'LineWidth', 2);
ylim([0 maxY]);
xlim([0 t(end)]);
title('Fourth Harmonic');
xlabel('t, s');
```




```
figure
subplot(2,1,1);
plot(t, Harm(5,:), 'LineWidth', 2);
ylim([0 maxY]);
xlim([0 t(end)]);
title('Fifth Harmonic');
xlabel('t, s');

subplot(2,1,2);
plot(t, Harm(6,:), 'LineWidth', 2);
ylim([0 maxY]);
xlim([0 t(end)]);
title('Sixth Harmonic');
xlabel('t, s');
```



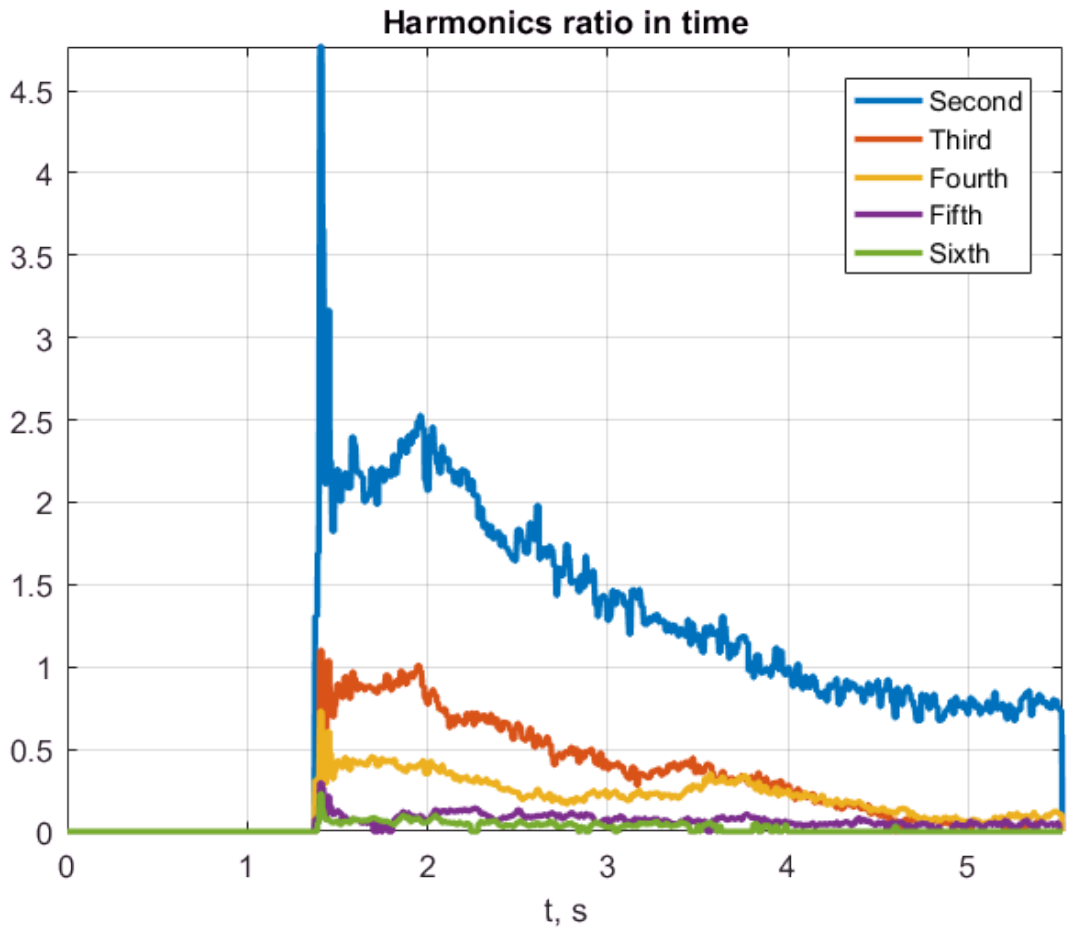

```
figure
subplot(1,1,1);
plot(t, Harm(1,:), 'LineWidth', 2);
hold on
plot(t, Harm(2,:), 'LineWidth', 2);
plot(t, Harm(3,:), 'LineWidth', 2);
plot(t, Harm(4,:), 'LineWidth', 2);
plot(t, Harm(5,:), 'LineWidth', 2);
plot(t, Harm(6,:), 'LineWidth', 2);
ylim([0 maxY]);
xlim([0 t(end)]);
grid on
title('Harmonics together');
xlabel('t, s');
legend('First', 'Second', 'Third', 'Fourth', 'Fifth', 'Sixth');
```



Покажу изменение во времени тембра, то есть отношения гармоник

```
for i = 1:length(Fref)-1
    for j = 1:N
        if Harm(i+1, j) < 7e-7 || Harm(1,j) < 7e-7
            Ar(i,j) = 0;
        else
            Ar(i,j) = Harm(i+1,j) ./ Harm(1,j);
        end
    end

end

maxY = max(max(Ar)');

figure
subplot(1,1,1);
plot(t, Ar(1,:), 'LineWidth', 2);
hold on
plot(t, Ar(2,:), 'LineWidth', 2);
plot(t, Ar(3,:), 'LineWidth', 2);
plot(t, Ar(4,:), 'LineWidth', 2);
plot(t, Ar(5,:), 'LineWidth', 2);
ylim([0 maxY]);
xlim([0 t(end)]);
grid on
title('Harmonics ratio in time');
xlabel('t, s');
```

```
legend('Second', 'Third', 'Fourth', 'Fifth', 'Sixth');
```



Можно сравнить полученные результаты отношения гармоник с полученными ранее для всей длительности сигнала. Временное проедставление отношения гармоник дает намного больше полезной информации о тембре

Обратный синтез с потерями фазы

Воссоздаю обратно звук как сумму этих гармоник. Звук должен отличаться от гитарной струны, но и чем то быть похожим

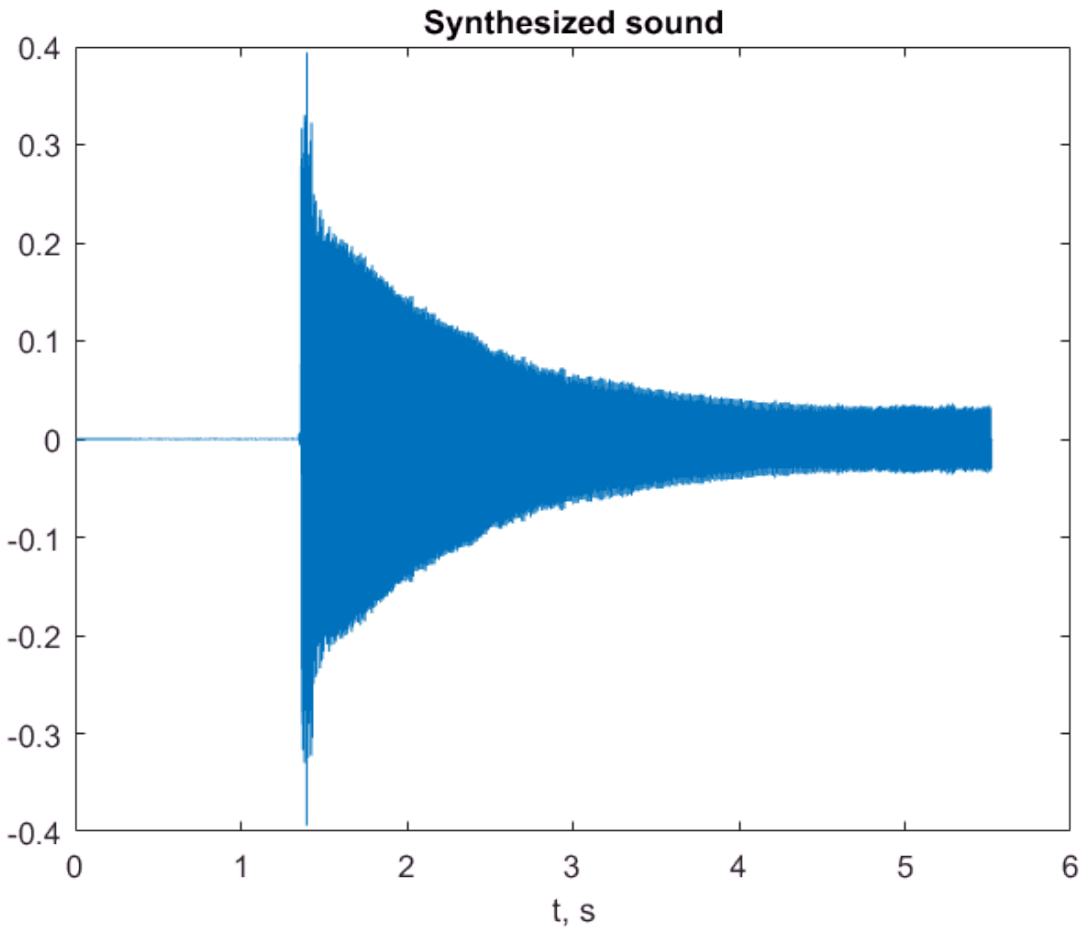
Синтез звука

```
NewSound = zeros(1,N);

for i = 1:length(Fref)
    NewSound = NewSound + Harm(i,:) .* sin(Fharm(i)*2*pi*t);
end
```

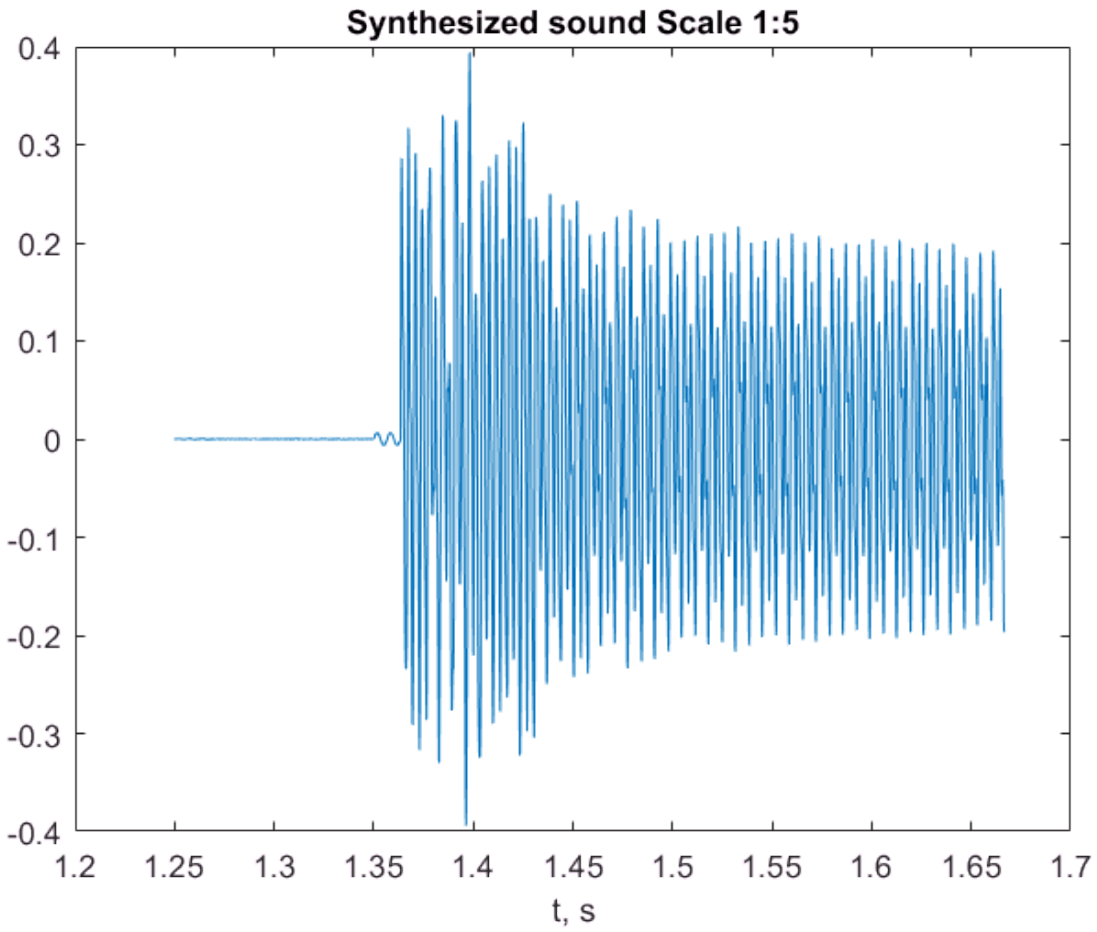
Gain

```
Gain = 1e3;
NewSound = NewSound .* Gain;

figure
plot(t, NewSound);
xlabel('t, s');
```

```
title('Synthesized sound');
```




```
figure
plot(t(75e2:1:1e4), NewSound(75e2:1:1e4));
xlabel('t, s');
title('Synthesized sound Scale 1:5');
```




```
sound(NewSound, fs);
```

Спектр синтезированного звука

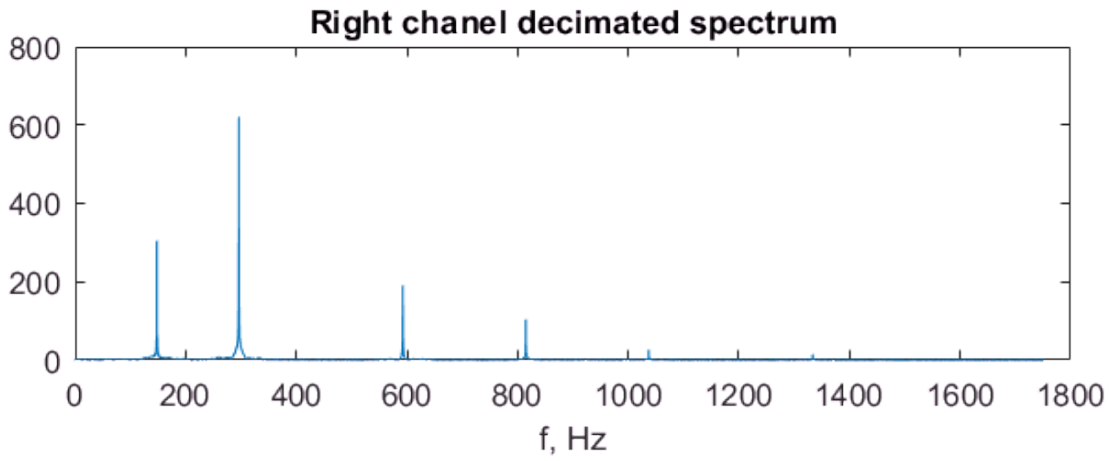
```
df = fs/N;
Fm = fs/2;
f = -Fm:df:Fm - df;

A  = abs(fftshift(fft(NewSound)));
Ne = N/2 + N * Fref(end)/Fm;
```

```
Ne = 2.6255e+04
```

```
figure
subplot(2,1,1);
plot(f(N/2 + 1:Ne), A(N/2 + 1:Ne));
```

```
Warning: Integer operands are required for colon operator when used as index
Warning: Integer operands are required for colon operator when used as index
```

```
title('Right chanel decimated spectrum');
xlabel('f, Hz');
```



Сохраняю результаты

```
filename = 'SynSound_02.06.2020_5.wav';
audiowrite(filename, NewSound, fs)
```

Обратный синтез с учетом фазы

При мат анализе выше использовался только правый канал звукоряда и при исследовании спектра использовался его модуль, чем была проигнорирована фаза. Повторю все действия без потери фазы и при обратном синтезе воссоздам спектр только из гармоник и получу синтезированный звук обратным преобразованием фурье.

(Rs - right spectrum)

```
Rs = zeros(Nfrag, Nw);
Ls = zeros(Nfrag, Nw);
Rsap = zeros(Nfrag, floor(Nw/2));
for i = 1:Nfrag
    ns = (i-1)*Nw + 1;
    ne = i*Nw;
    Rs(i,:)  = fft(x(ns:ne,1));
    Ls(i,:)  = fft(x(ns:ne,2));
end
```

```
Undefined function 'x' for input arguments of type 'double'.
```

Собираю временные зависимости каждой из гармоник каждого канала в комплексном виде, не упуская фазу

```
HarmR = zeros(4, N);
HarmL = zeros(4, N);
Fref = [146 293 440 586];
f    = -fs/2:df:fs/2 - df;

for i = 1:Nfrag
    [pksR, FharmR] = findpeaks(fftshift(abs(Rs(i,:))), f, 'MinPeakDistance', 60);
    [pksL, FharmL] = findpeaks(fftshift(abs(Ls(i,:))), f, 'MinPeakDistance', 60);

    for j = 1:length(Fharm)
        for z = 1:length(Fref)

            if length(FharmR) > j-1
                if abs(FharmR(j) - Fref(z)) < 60
                    ns = (i-1)*Nw + 1;
                    ne = i*Nw;
                    Nr = floor(Nw * FharmR(j)/fs + 1 + Nw/2);
                    HarmR(z, ns:ne) = ones(1,Nw) * Rs(i,Nr);
                end
            end

            if length(FharmL) > j-1
                if abs(FharmL(j) - Fref(z)) < 60
                    ns = (i-1)*Nw + 1;
                    ne = i*Nw;
                    Nl = floor(Nw * FharmL(j)/fs + 1 + Nw/2);
                    HarmL(z, ns:ne) = ones(1,Nw) * Ls(i,Nl);
                end
            end

        end
    end
end
```

На каждое временное окно наблюдения создаю свой спектр, получаю матрицу спектров для каждого промежутка времени. После чего получу звукоряд для каждого промежутка и объединю его в один вектор звукоряда

Не использовать код. Зависает компьютер!!!

```
SpecR  = zeros(Nfrag, Nw);
SpecL  = zeros(Nfrag, Nw);

for i = 1:Nfrag
    for j = 1:Nw
        for z = 1:4

            Nf    = floor(Nw * FharmL(z)/fs + 1 + Nw/2);
            Nharm = floor(j + (i-1) * Nw)
            SpecR(Nf)      = SpecR(Nf) + HarmR(z, Nharm);
            SpecR(Nw - Nf) = SpecR(Nf);

            SpecL(Nf)      = SpecL(Nf) + HarmL(z, Nharm);
            SpecL(Nw - Nf) = SpecL(Nf);
```

```
        end
    end
end
```

```
Nharm = 1
Subscript indices must either be real positive integers or logicals.
```

Получаю левый и правый канал

```
SoundR = ifft(fftshift(SpecR));
SoundL = ifft(fftshift(SpecL));

figure
plot(t, SoundR);
xlabel('t, s');
title('Synthesized sound');

figure
plot(t(8e4:13e4), SoundR(8e4:13e4));
xlabel('t, s');
title('Synthesized sound Scale 1:5');

figure
plot(t(85e3:95e3), SoundR(85e3:95e3));
xlabel('t, s');
title('Synthesized sound Scale 1:27');

Sound  = [SoundR; SoundL]
```

3. Изменение тембра методом апроксимации кубическими сплайнами

Повторю нахождение временных зависимостей амплитуд гармоник, найденых во 2 пункте и вместо прямого их использования в качестве амплитуд гармонических колебаний сперва интерполирую их сплайнами тем самым повысив частоту дискретизации и, возможно, улучшив качество звука.