Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

# Лабораторная работа №7

Частотно-временной анализ нестационарных сигналов методом Гильберта - Хуанга.

Выполнил
студент гр. в3530904/00030 В.С. Баганов

Руководитель
доцент, к.т.н. В.С. Тутыгин

«____»_____ 202__ г.

Санкт-Петербург
2023

# Содержание

# 1. Частотно-временной анализ нестационарных сигналов методом Гильберта-Хуанга.

# 2. Цель работы

Цель данной работы — определить оптимальное количество удаляемых функций IMF при очистке зашумленного сигнала с помощью преобразования Хуанга в зависимости от вида сигнала и уровня шума на входе.

# 3. Программа работы

1. Измерить коэффициент подавления шума (KSKO) для стационарного гармонического сигнала для уровней шума 0.1–0.3 (шаг 0.1) при количестве удаляемых шумовых функций от 1 до 3. Для каждой комбинации уровня шума и количества удаляемых функций проводить испытания 10 раз (всего 90 измерений). Результаты этих 10 измерений занести в протокол. Из 10 значений выбрать наименьшее и наибольшее. Результат также зафиксировать в протоколе. Выбрать оптимальное количество исключаемых функций IMF для каждого уровня шума.

2. Повторить действия из п. 1 для нестационарного сигнала.

3. Применить преобразование Хуанга для очистки от шумов реального сигнала. Определить оптимальное количество исключаемых функций IMF.

# 4. Результаты работы

## 4.1. Обработка и анализ стационарного гармонического сигнала

Ниже приведен протокол испытаний для стационарного гармонического сигнала:

| | KSKO | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| NF/N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 5,6518 | 6,0781 | 5,565 | 6,3869 | 5,9324 | 6,0771 | 5,8007 | 5,7233 | 5,7186 | 5,6143 |
| 2 | 8,2282 | 0,0577 | 8,1166 | 7,6188 | 3,6118 | 3,275 | 8,7481 | 9,0356 | 8,3574 | 7,7705 |
| 3 | 9,7522 | 0,3841 | 5,1526 | 0,0387 | 1,953 | 3,6416 | 2,1295 | 2,3385 | 4,5046 | 4,0267 |
| 1 | 5,8229 | 5,9612 | 5,9067 | 6,241 | 5,9517 | 5,9189 | 5,9189 | 6,0545 | 5,9616 | 5,6199 |
| 2 | 7,8547 | 9,0842 | 8,0444 | 8,5935 | 8,4831 | 7,5918 | 8,2324 | 8,5626 | 8,7795 | 8,5694 |
| 3 | 9,9618 | 6,482 | 11,0581 | 13,2041 | 10,4983 | 10,1714 | 5,0209 | 11,2806 | 8,9203 | 8,098 |
| 1 | 5,9139 | 6,0375 | 6,1055 | 6,0555 | 5,7782 | 5,9945 | 6,3082 | 5,5796 | 5,8371 | 6,0319 |
| 2 | 8,8498 | 9,1759 | 8,2295 | 8,2677 | 8,5919 | 7,8915 | 9,0231 | 8,1482 | 7,7644 | 8,4905 |
| 3 | 8,759 | 8,8208 | 8,1492 | 10,1538 | 9,0303 | 8,5114 | 8,37 | 8,5153 | 9,344 | 8,2969 |

Q - СКО шума на входе

NF - кол-во исключаемых функций IMF

KSKO - коэффициент подавления шума по результатам 10 повторных измерений

N - номер повторного измерения

| Q | NF | KSKO | |
|---|---|---|---|
| | | от | до |
| 0,1 | 1 | 5,565 | 6,3869 |
| | 2 | 0,0577 | 9,0356 |
| | 3 | 0,0387 | 9,7522 |
| 0,2 | 1 | 5,6199 | 6,241 |
| | 2 | 7,5918 | 9,0842 |
| | 3 | 5,0209 | 13,204 |
| 0,3 | 1 | 5,5796 | 6,3082 |
| | 2 | 7,7644 | 9,1759 |
| | 3 | 8,1492 | 10,154 |

Из полученных данных и таблицы выше видно.

Немого забегая вперед для оптимального значения погрешности (чтобы сигнал был хорошо отфильтрован) нужно чтобы в среднем KSKO приближалось или в среднем было равно от 8 до 10

### 4.1.1. Зашумление Q=0.1

При низком уровне шума стабильные результаты возможны только при NF=1 т.е. при вычитании 1-ой функции шума. Коэффициент подавления шума при этом колеблется от 5.7447 до 6.2547. При NF=2 и NF=3 очень большой разброс значений (NF=2, от 0.0577 до 9.0356) и (NF=3, от 0.0387 до 9.7522), который говорит, что вычитанib 2-х или 3-х шумовых функции исходный сигнал деформируется т.е мы теряем полезные данные вычитая так много шумовых функций. Ниже на графике видно что наиболее

похожая функция при NF=1, остальные деформированны ( у NF=3 в конце появилость искажение).

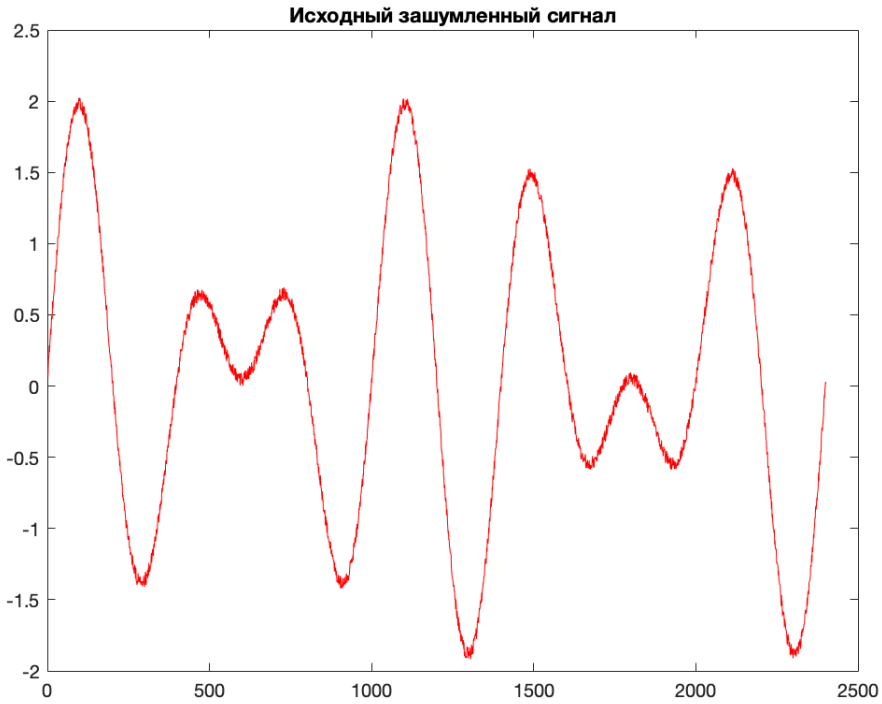



Рисунок 4.1. Исходный зашумленный сигнал Q=0.1

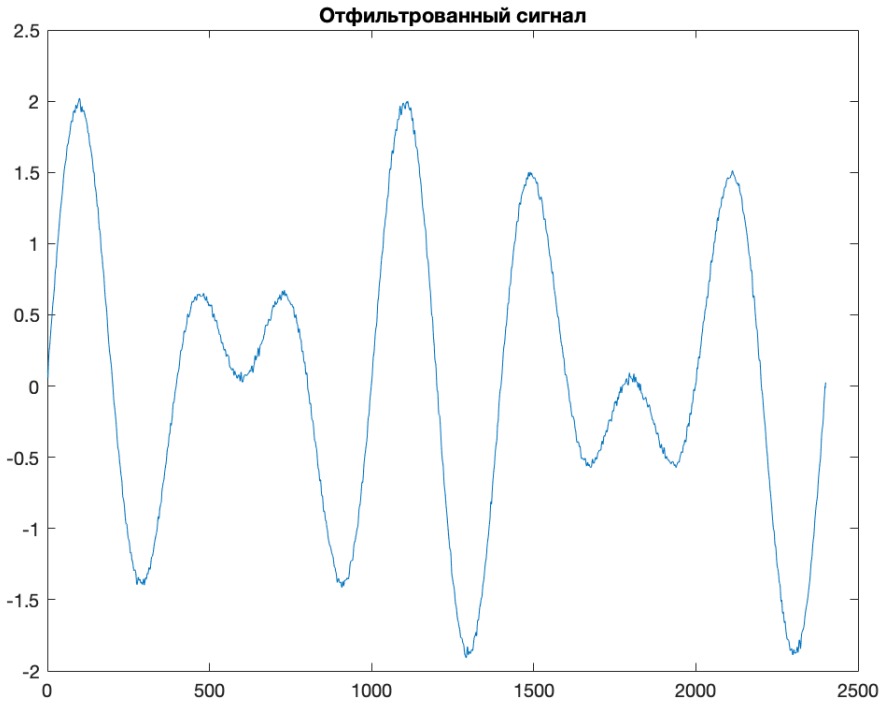



Рисунок 4.2. Q=0.1, NF=1

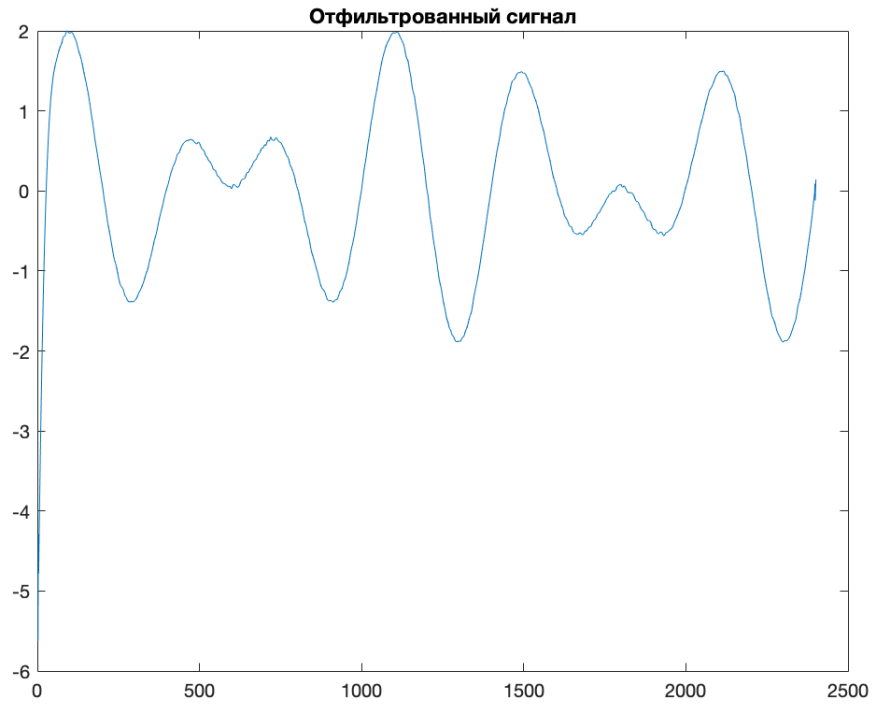



Рисунок 4.3. Q=0.1, NF=2

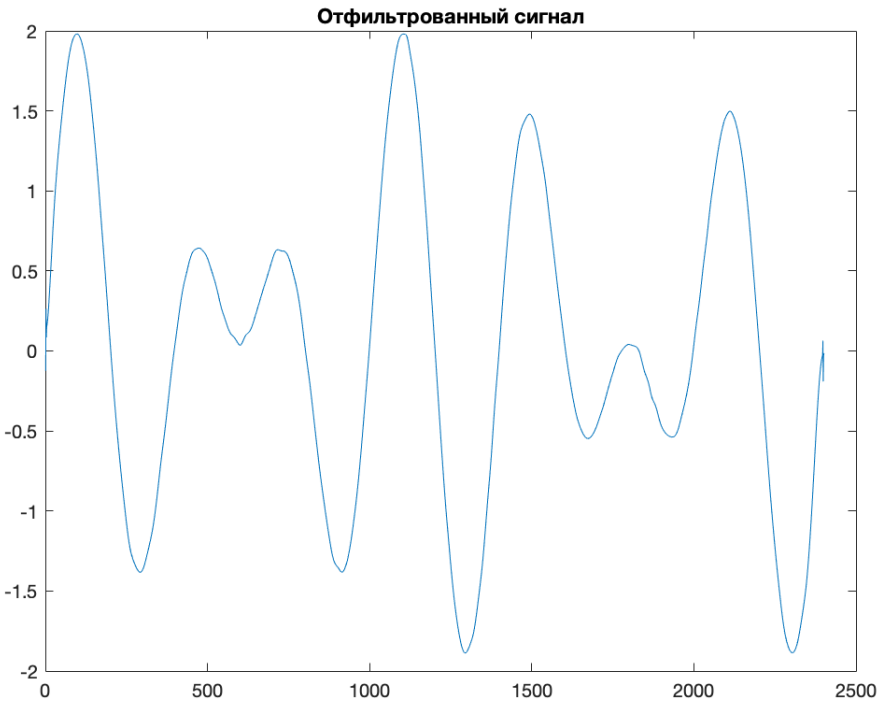



Рисунок 4.4. Q=0.1, NF=3

### 4.1.2. Зашумление Q=0.2

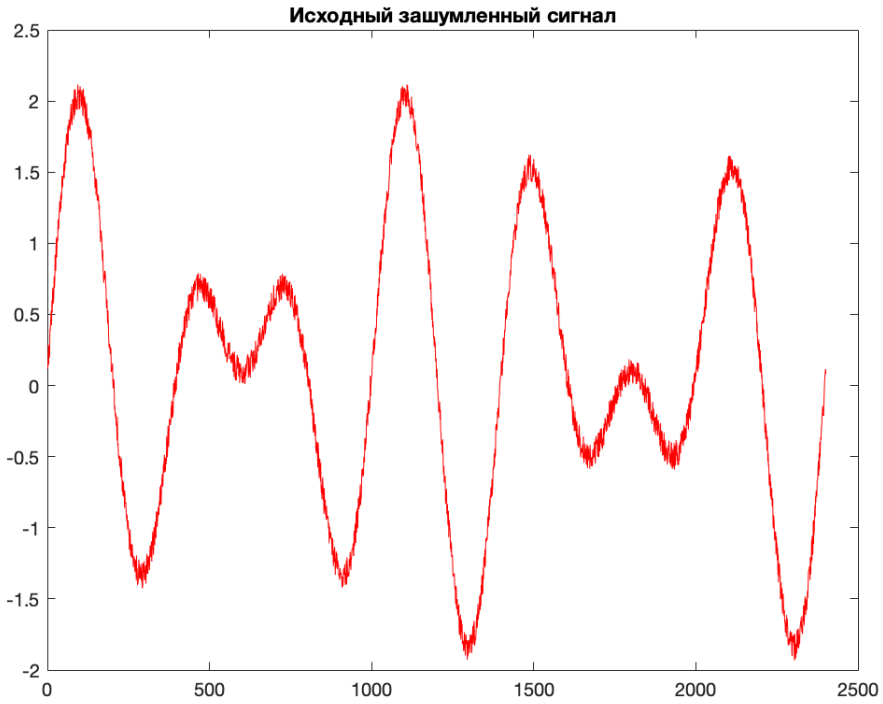



Рисунок 4.5. Исходный зашумленный сигнал Q=0.2

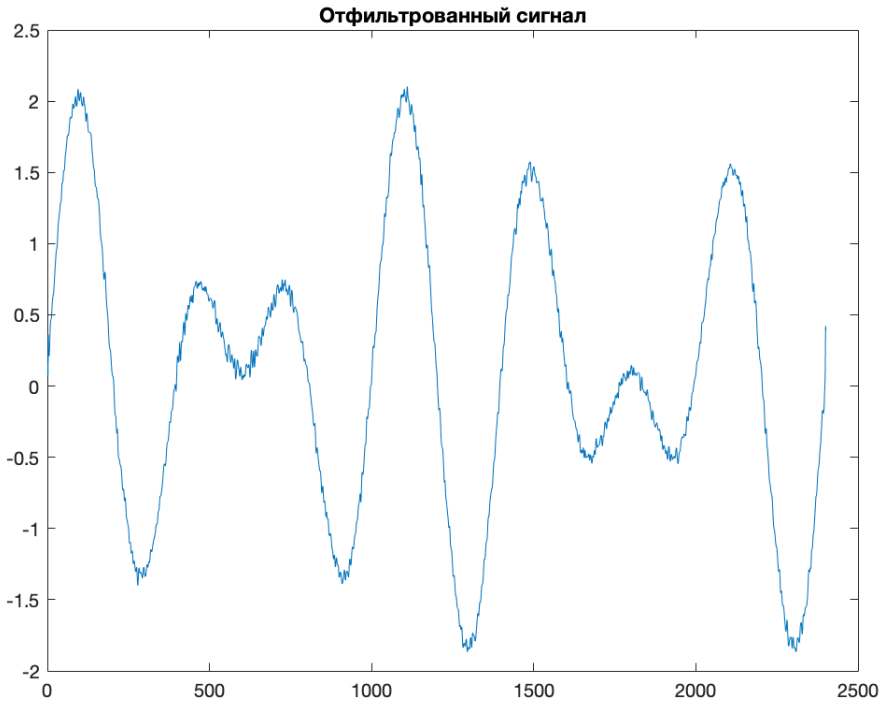



Рисунок 4.6. Q=0.2, NF=1

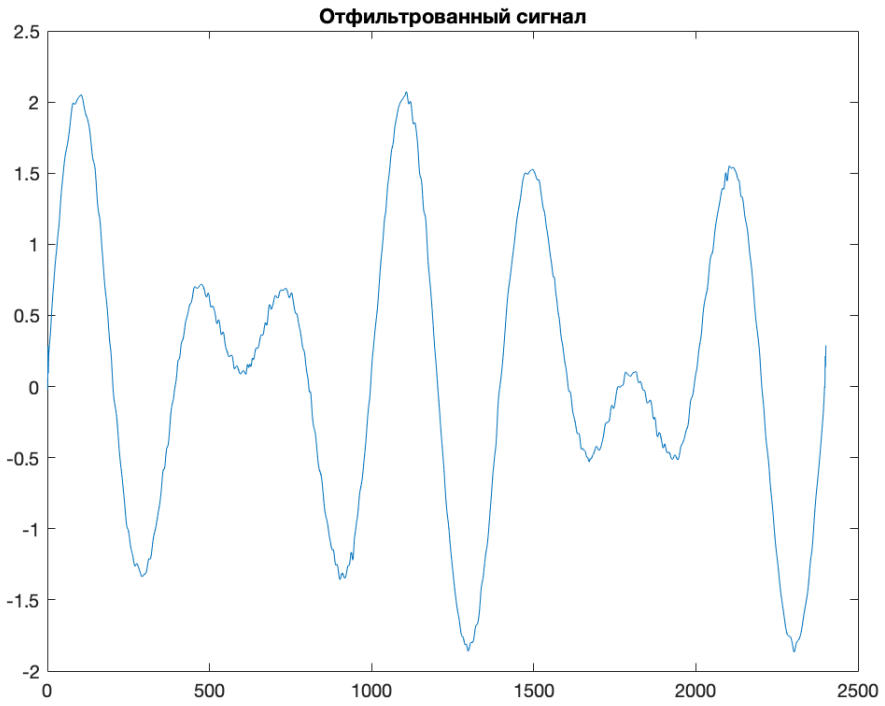



Рисунок 4.7. Q=0.2, NF=2

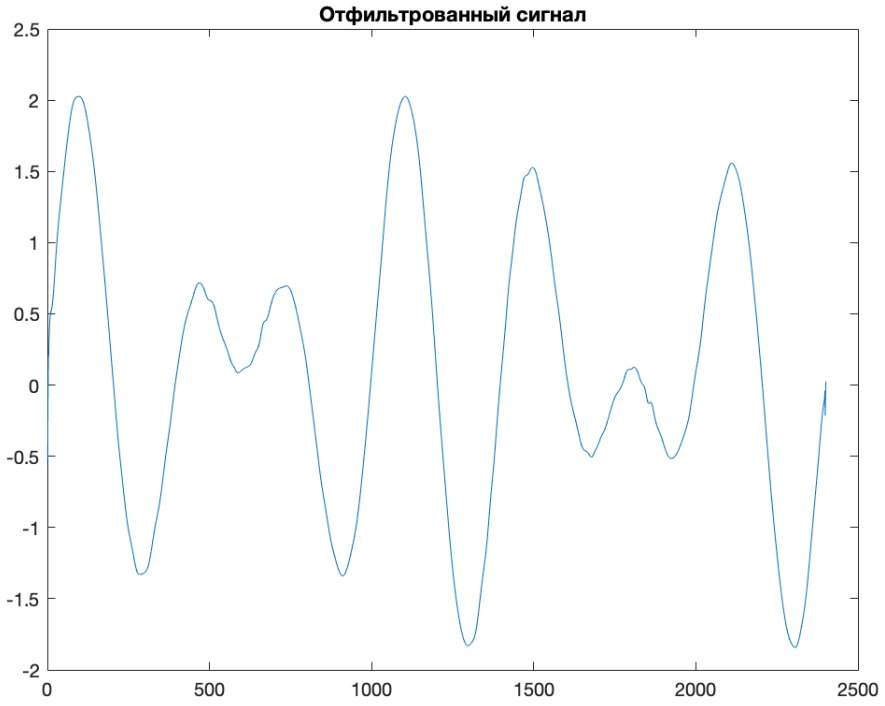



Рисунок 4.8. Q=0.2, NF=3

Видно что при (Q=0.2, NF=2, от 7.5918 до 9.0842) сигнал имеет уже практически исходный (хоть и небольшой шум остался), но все же более очищенный ситнал наблюдается

при (Q=0.2, NF=3, от 5.0209 до 13.204).

### 4.1.3. Зашумление Q=0.3

При сильном зашумлении сигнала (Q=0.3)

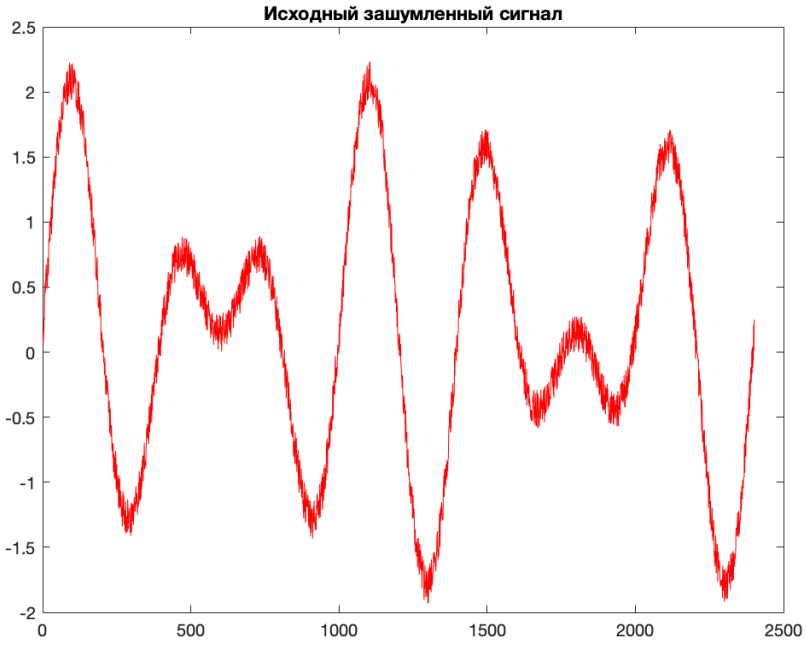



Рисунок 4.9. Исходный зашумленный сигнал Q=0.1

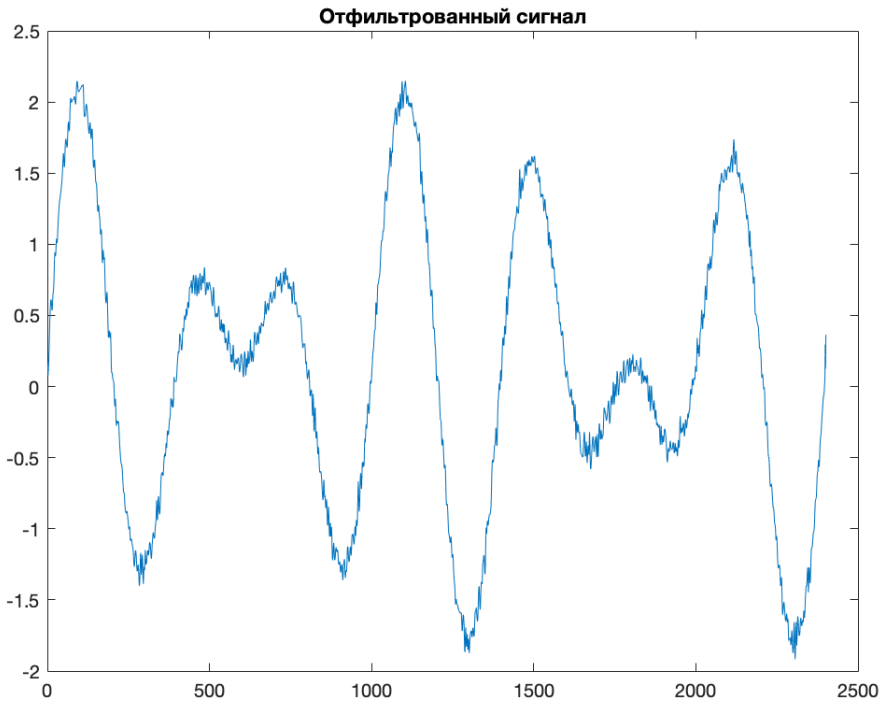



Рисунок 4.10. Q=0.3, NF=1

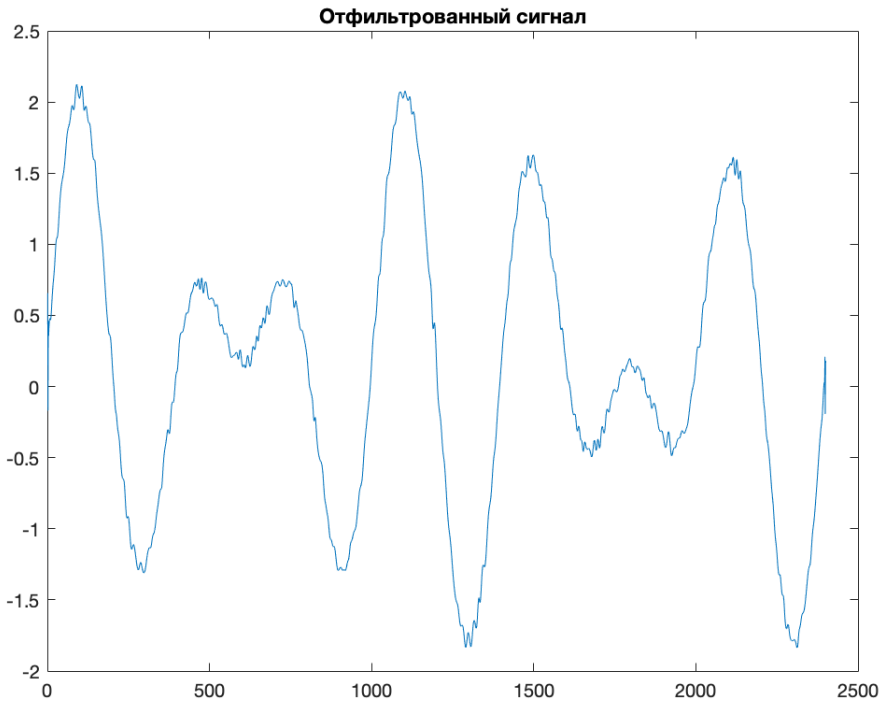



Рисунок 4.11. Q=0.3, NF=2

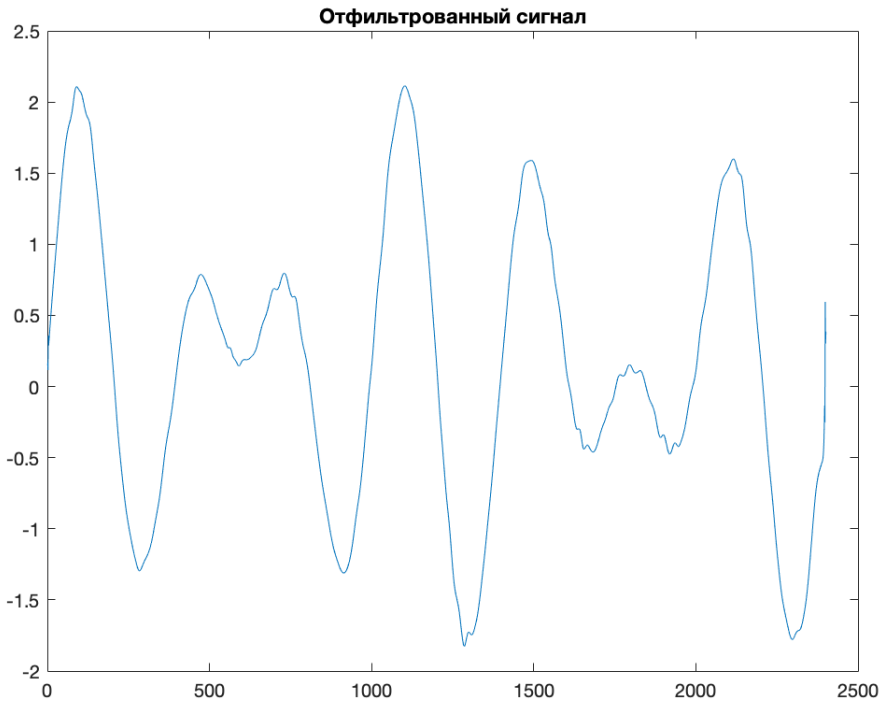



Рисунок 4.12. Q=0.3, NF=3

Видно что максимально точное представление сигнла из всех при (Q=0.3, NF=3) от 8.1492 до 10.154),

хотя при (Q=0.3, NF=2) тоже может быть использован, так как разброс значений не такой сильный (Q=0.3, NF=2, от 7.7644 до 9.1759) но видно что в сигнале присутствует шум.

Общий ввывод при стационарном гармоническом сигнале: при малом зашумлении нужно вычитать 1 функцию шума, а при Q=0.2 и Q=0.3 лучше 3 функции шума.

## 4.2. Обработка и анализ нестационарного сигнала

Ниже приведен протокол испытаний для нестационарного сигнала:

| | KSKO | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| NF/N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 5,9895 | 5,7447 | 5,7961 | 6,2547 | 6,0006 | 5,8679 | 5,9905 | 5,761 | 5,971 | 5,948 |
| 2 | 8,4946 | 9,2425 | 8,1351 | 8,8152 | 9,2207 | 8,4933 | 9,0035 | 9,325 | 9,0833 | 8,716 |
| 3 | 10,599 | 8,6232 | 10,3477 | 8,5326 | 10,5976 | 11,8784 | 11,746 | 12,6008 | 8,8328 | 8,741 |
| 1 | 6,1231 | 5,9031 | 6,1004 | 6,1392 | 5,9884 | 5,9187 | 5,7515 | 6,0559 | 5,9847 | 6,079 |
| 2 | 5,2388 | 7,918 | 8,911 | 9,2696 | 8,6558 | 8,6021 | 8,8431 | 8,1754 | 8,7038 | 9,129 |
| 3 | 10,649 | 11,5441 | 2,5474 | 12,0392 | 11,041 | 9,9169 | 9,0397 | 11,1883 | 10,944 | 8,401 |
| 1 | 6,2681 | 5,7904 | 5,755 | 6,1379 | 6,0984 | 5,8339 | 6,0542 | 6,2001 | 5,9202 | 6,035 |
| 2 | 8,2726 | 9,0927 | 8,5411 | 8,8626 | 8,8805 | 8,292 | 8,3233 | 8,2557 | 8,3478 | 8,624 |
| 3 | 7,94 | 11,2085 | 11,8188 | 7,2751 | 10,4332 | 8,7877 | 11,969 | 12,8898 | 9,0053 | 12,36 |

Q - СКО шума на входе

NF - кол-во исключаемых функций IMF

KSKO - коэффициент подавления шума по результатам 10 повторных измерений

N - номер повторного измерения

| Q | NF | KSKO | |
|---|---|---|---|
| | | от | до |
| 0,1 | 1 | 5,7447 | 6,2547 |
| | 2 | 8,1351 | 9,325 |
| | 3 | 8,5326 | 12,601 |
| 0,2 | 1 | 5,7515 | 6,1392 |
| | 2 | 5,2388 | 9,2696 |
| | 3 | 2,5474 | 12,039 |
| 0,3 | 1 | 5,755 | 6,2681 |
| | 2 | 8,2557 | 9,0927 |
| | 3 | 7,2751 | 12,89 |

Из полученных данных и таблицы выше, видно что при нестационарном сигнале мы получили более стабильные результатты KSKO.

Немого забегая вперед для оптимального значения погрешности (чтобы сигнал был хорошо отфильтрован) нужно, чтобы в среднем KSKO приближалось или в среднем было равно от 8 до 10.

### 4.2.1. Зашумление Q=0.1

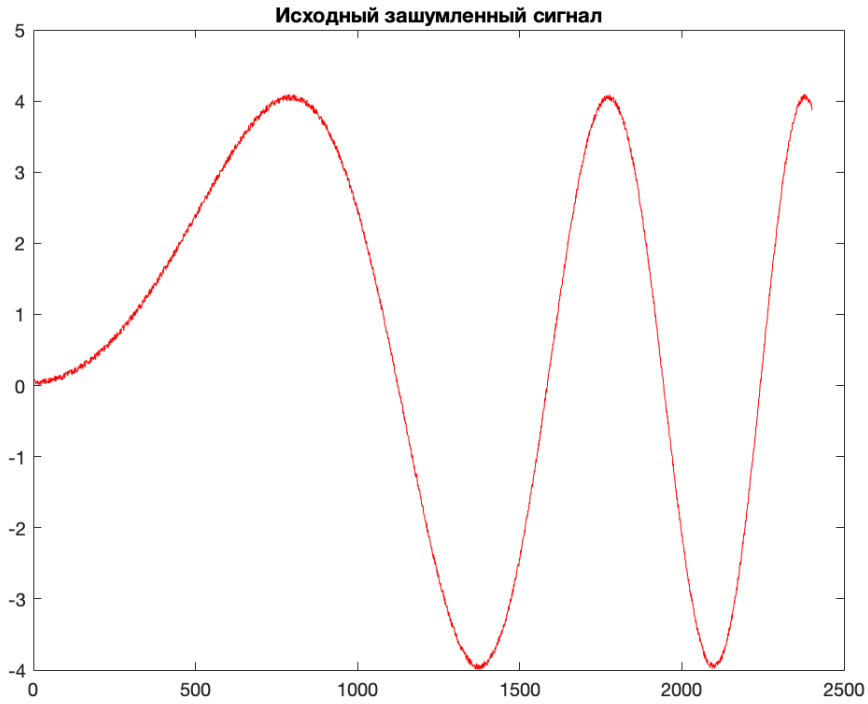



Рисунок 4.13. Исходный зашумленный сигнал Q=0.1

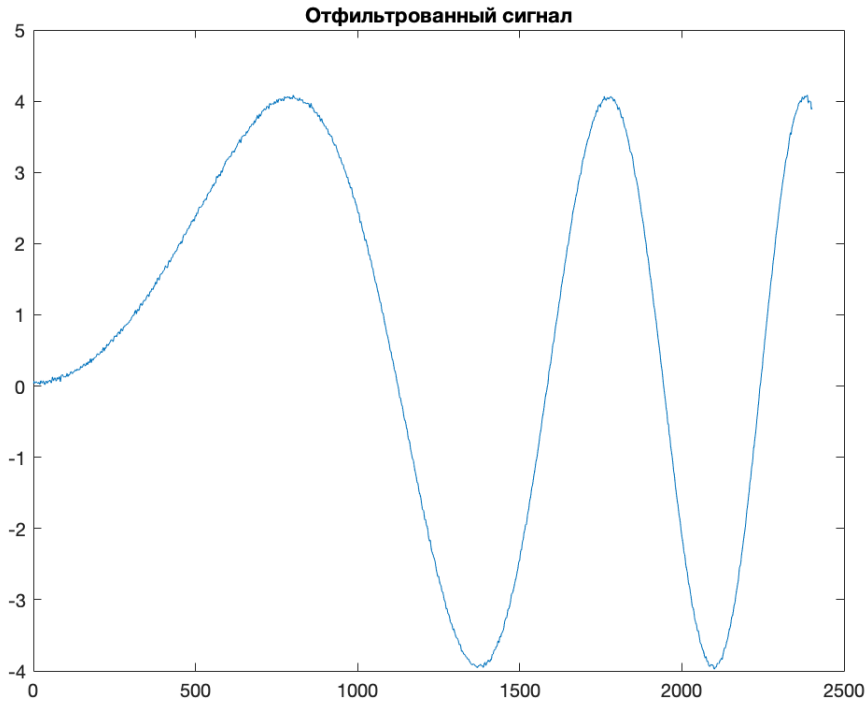



Рисунок 4.14. Q=0.1, NF=1

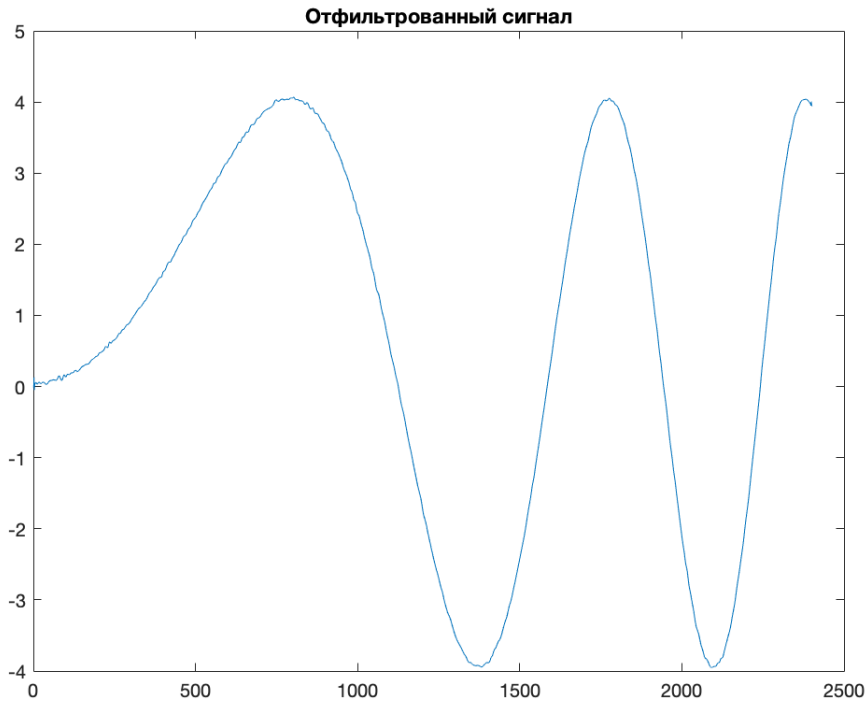



Рисунок 4.15. Q=0.1, NF=2

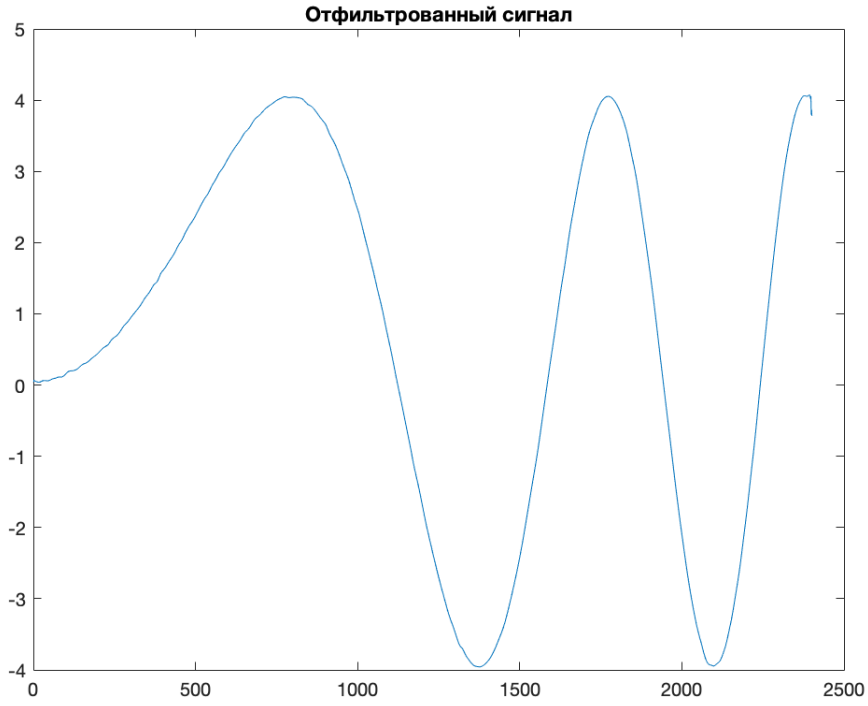



Рисунок 4.16. Q=0.1, NF=3

При зашумлении Q=0.1 нестационарных сигналов, чем больше вычитаем функций шума NF=3, тем лучше отфильтрован сигнал.

### 4.2.2. Зашумление Q=0.2

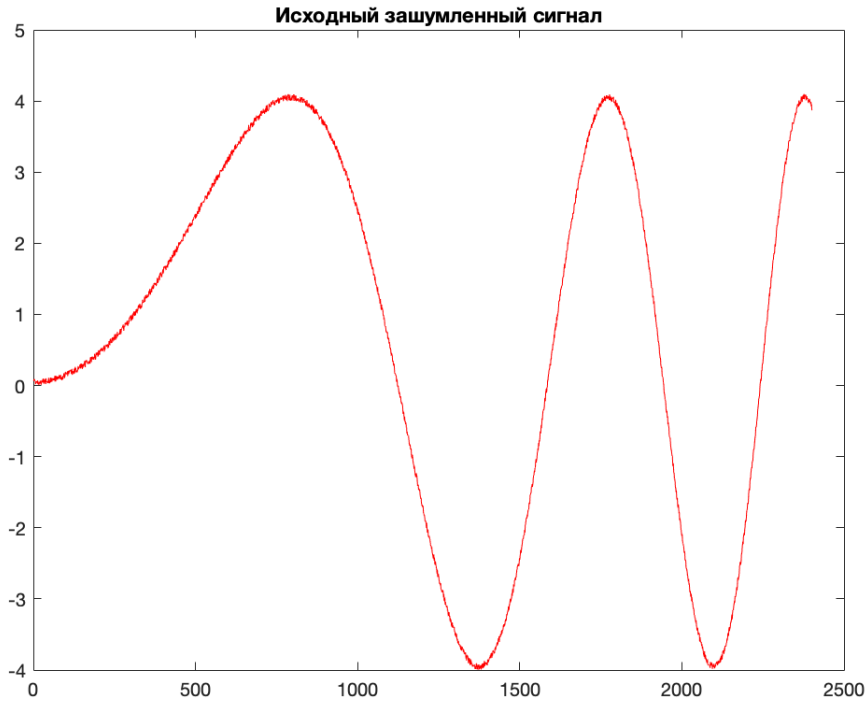



Рисунок 4.17. Исходный зашумленный сигнал Q=0.2

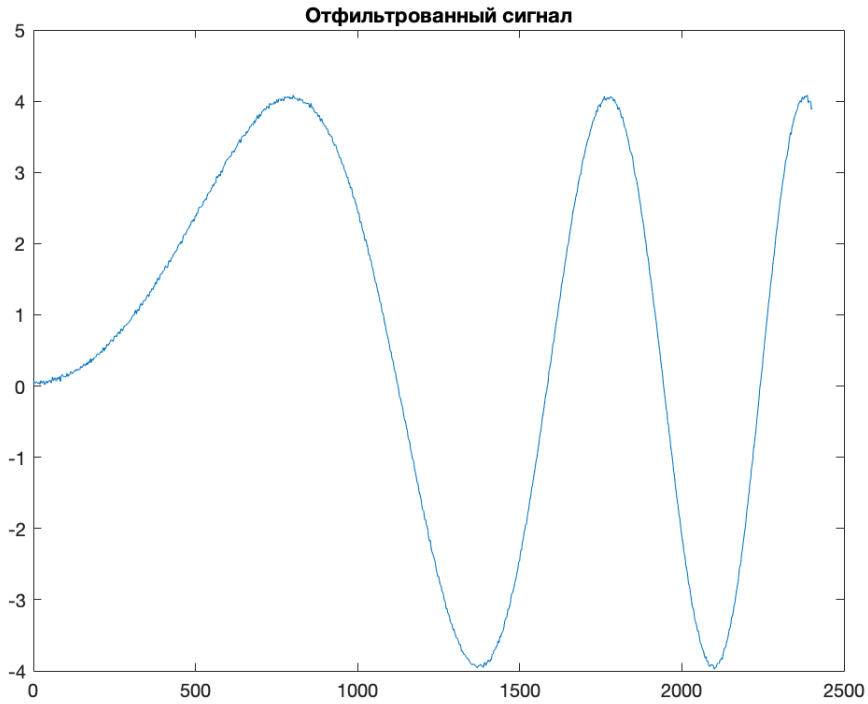



Рисунок 4.18. Q=0.2, NF=1

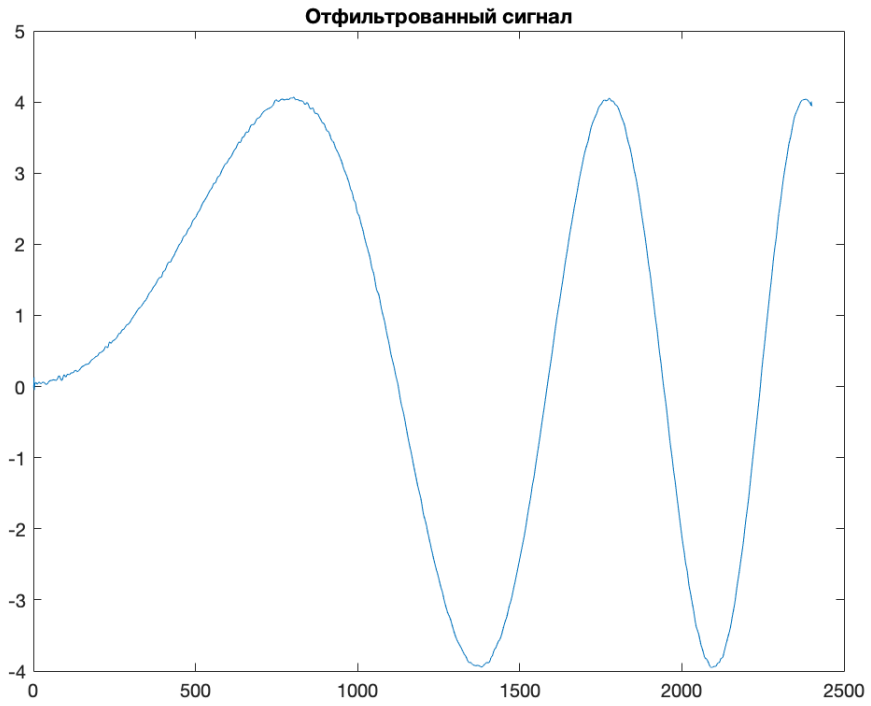



Рисунок 4.19. Q=0.2, NF=2

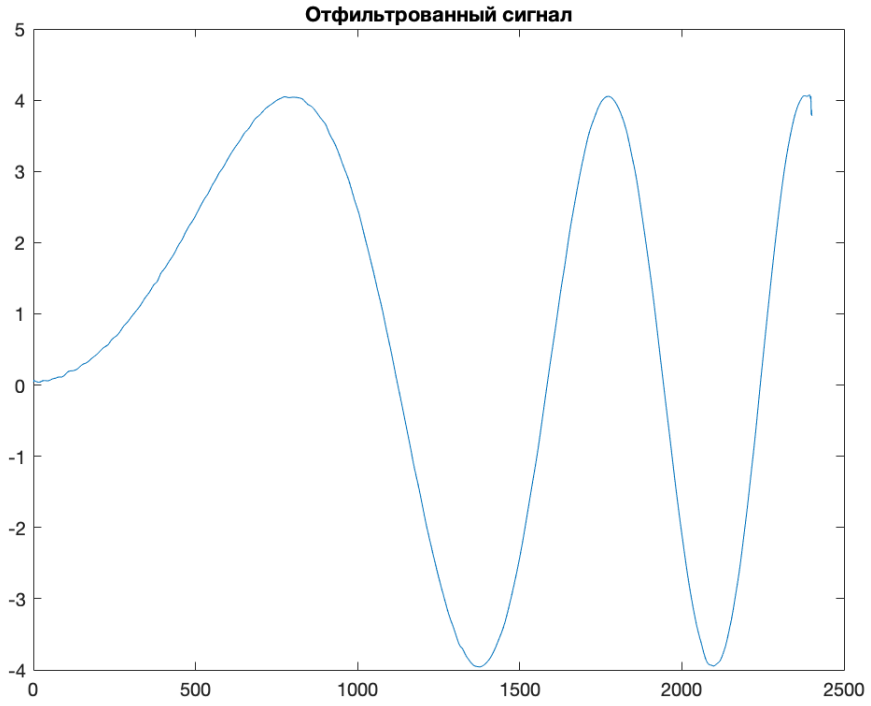



Рисунок 4.20. Q=0.2, NF=3

При зашумлении Q=0.2 нестационарных сигналов, чем больше вычитаем функций шума NF=3, тем лучше отфильтрован сигнал.

### 4.2.3. Зашумление Q=0.3

При сильном зашумлении сигнала (Q=0.3)

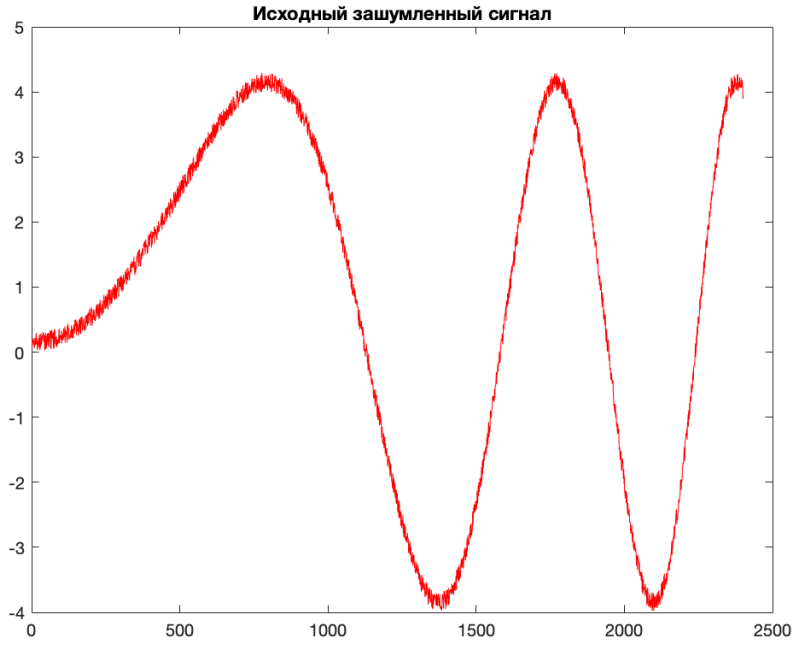



Рисунок 4.21. Исходный зашумленный сигнал Q=0.3

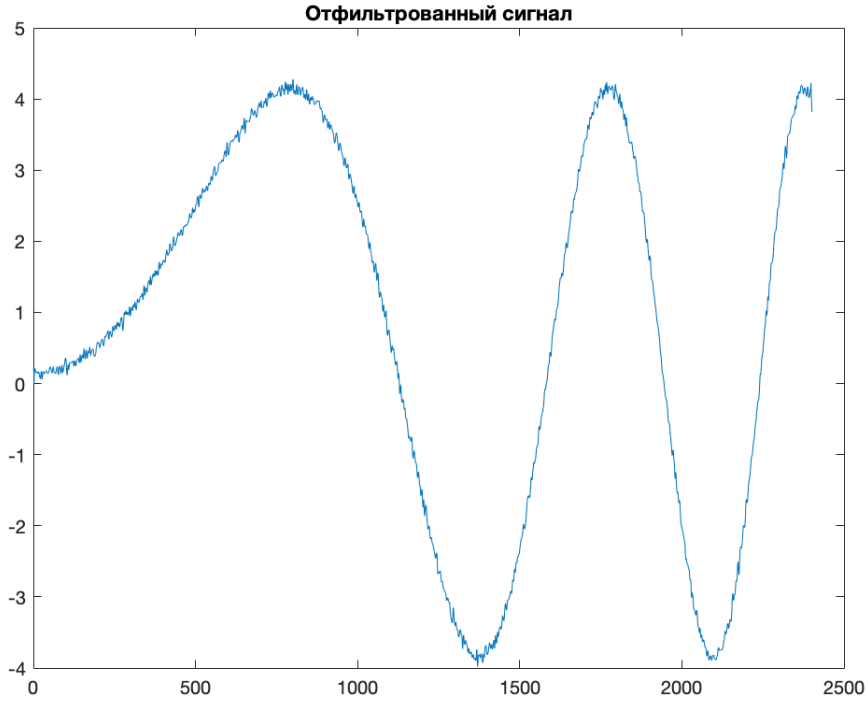



Рисунок 4.22. Q=0.3, NF=1

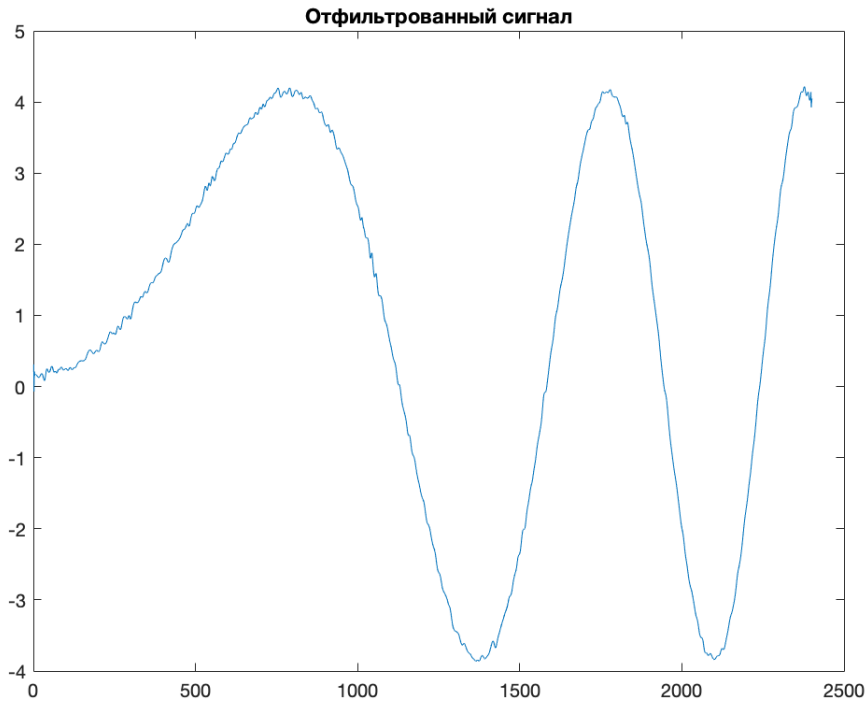



Рисунок 4.23. Q=0.3, NF=2

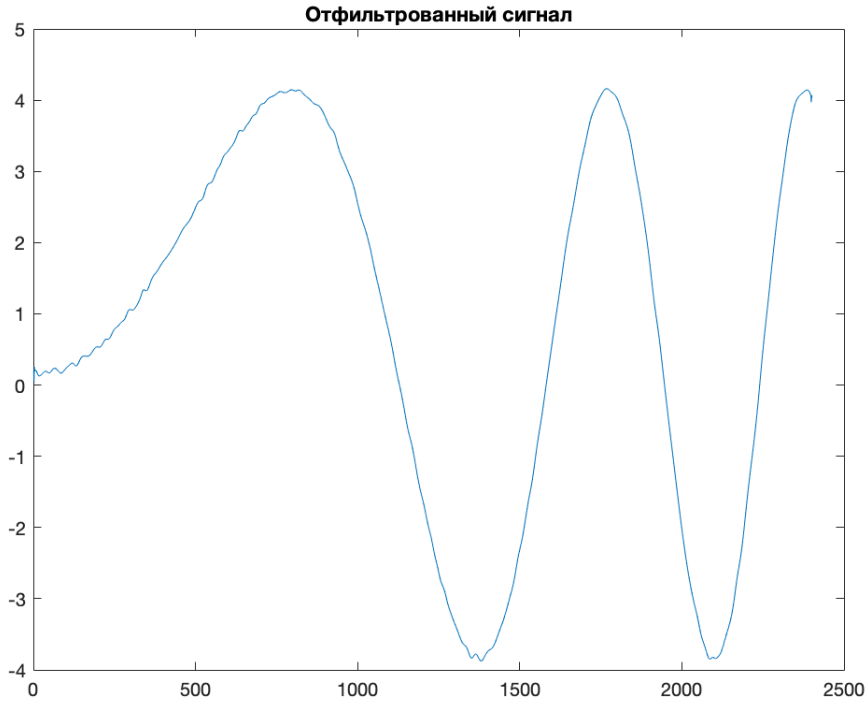
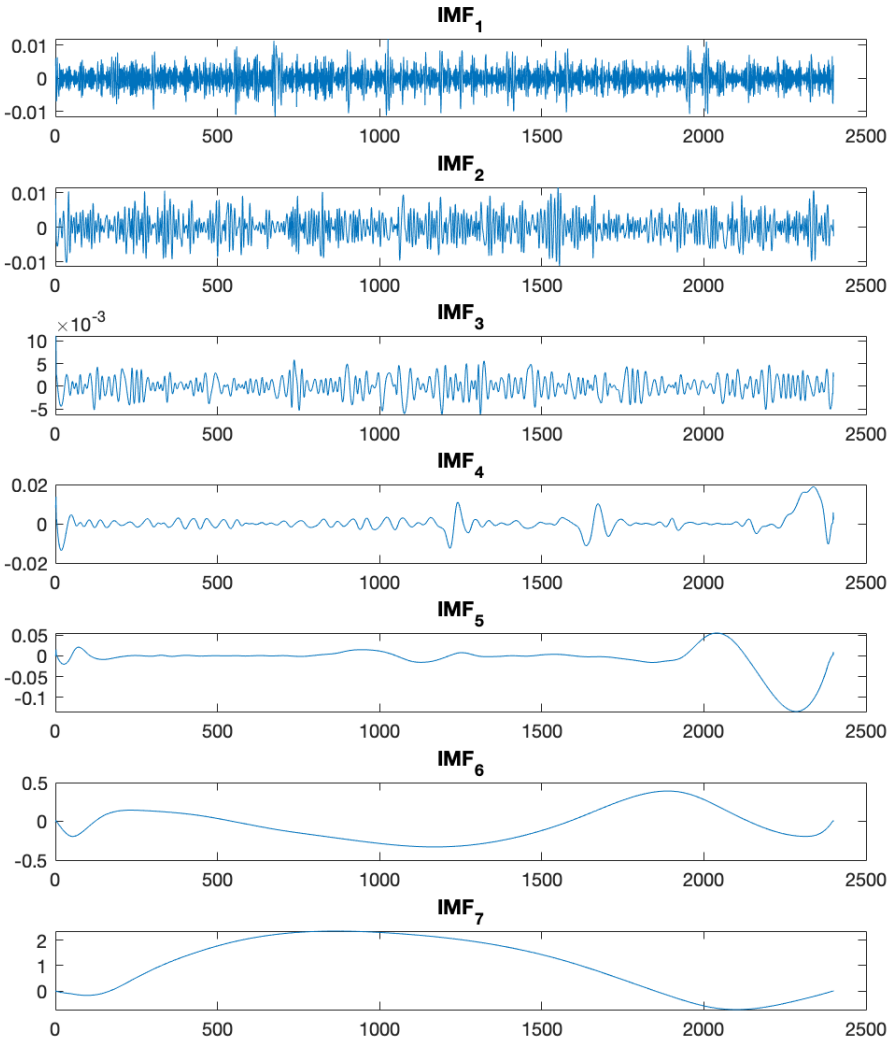



Рисунок 4.24. Q=0.3, NF=3

При большом зашумлении Q=0.3 нестационарных сигналов, чем больше вычитаем функций шума NF=3, тем лучше отфильтрован сигнал.

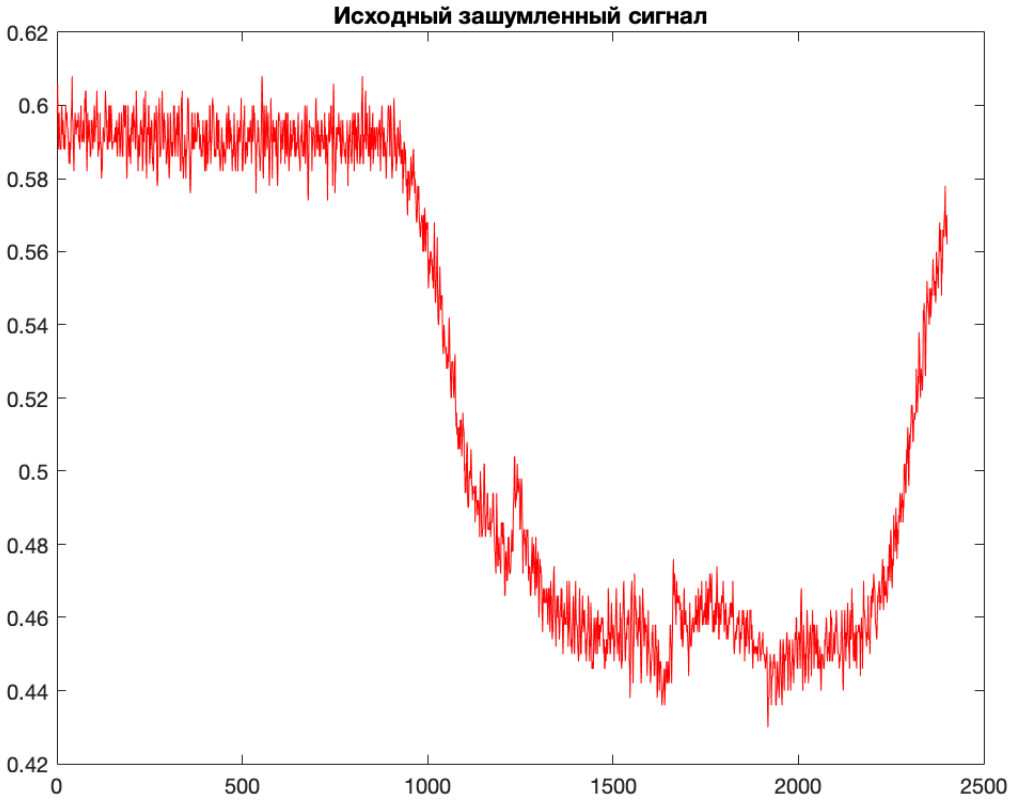
## 4.3. Обработка и анализ реального сигнала

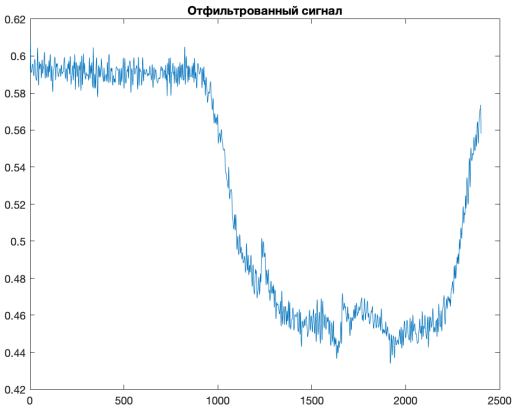
Обычно при разложении сигнала, низкочастотная составляющая несёт в себе информацию об исходном сигнале. Высокочастотная — это шум и мелкие детали. На лекции так же было сказано, что когда IMF_ приобретает значения около 0.005 (тысчных занчений), то ниже уже будут низкочастотные составляющие, а именно функции исходного сигнала. На рисунке ниже видно, что до IMF_3 идут шумовые функции, а после IMF_3 идут функции чистого сигнала.

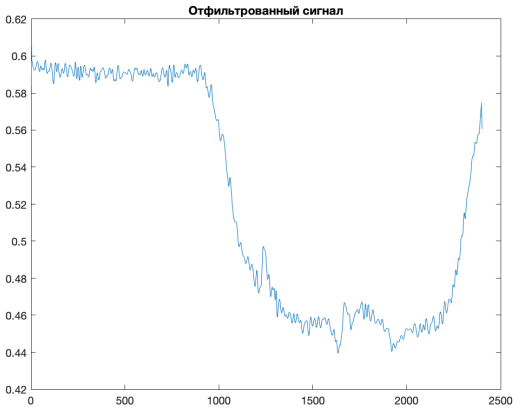


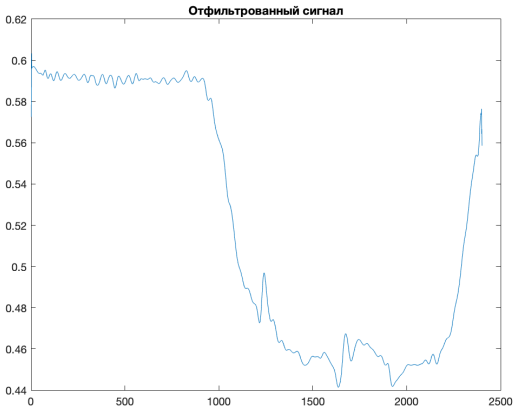

Ниже приведен график реального сигнала при сильном зашумлении (Q=0,3 красный). Гафики отфильтрованного сигнала синим цветом. На граифках видно как вычитая функциию шума 1,2 и 3 мы получаем более четкий сигнал. Так как мы не знаем уро-

вень зашумеленности сигнала (а тем более самого чистого сигнала) мы можем только визуально отобрать максимально близкий график сигнла очишенный от шума. На последней картинке изображен предположительно исходный сигнал при NF=3 (удаления 3-х функций содержащих шум).




Отфильтрованный сигнал
0.62
0.6
0.58
0.56
0.54
0.52
0.5
0.48
0.46
0.44
0.42
0
500
1000
1500
2000
2500


Отфильтрованный сигнал
0.62
0.6
0.58
0.56
0.54
0.52
0.5
0.48
0.46
0.44
0.42
0
500
1000
1500
2000
2500


Отфильтрованный сигнал
0.62
0.6
0.58
0.56
0.54
0.52
0.5
0.48
0.46
0.44
0
500
1000
1500
2000
2500

# 5. Вывод

Эмпирическую модовую декомпозицию Хуанга можно использовать для очистки сигналов от шумов.

Из анализа данных становится понятно, что чем больше зашумлен сигнал, тем точнее можно вычислить шум и исключить его из исходного сигнала.игнала.

Каждая IMF содержит более низкие частотные составляющие, чем извлеченная перед ней.

По мере увеличения количества функций IMF относительная среднеквадратическая погрешность реконструкции достаточно сложных и протяженных сигналов уменьшается, но, как правило, имеет определенный минимум.

Количество функций IMF в различных реализациях случайного сигнала изменяется от 8 до 14.

Процесс снижения погрешности при увеличении количества функций IMF приведен ниже на рисунке:


Преобразование Гильберта – Хуанга является эффективным методом обработки нестационарных сигналов. Благодаря наличию таких важнейших свойств, как локальность и адаптивность, появляется возможность более точно выявлять скрытые в шумах амплитудные и частотные модуляции, выполнять идентификацию частотных и временных диапазонов.

# 6. Листинг Matlab. lab7_2020.m

```
%Эмпирическая модовая декомпозиция Хуанга.
 %Используется для  очистки сигналов от шумов

clc;%очистка Command Window
N=2401;%количество точек
d=0.005;%шаг изменения переменной time
M=(N-1)*d;%
time = 0:d:M;
Q=0.3;%уровень шума в долях СКО
NF=1;%кол-во функций разложения, содержащих шум (NF=1..3)
KP1=5;%кол-во периодов первого сигнала
KP2=7;%кол-во периодов второго сигнала

 %ПРОГРАММНАЯ ГЕНЕРАЦИЯ СИГНАЛА
  noise=rand(N);%генерация массива равномерно распределенного шума
% noise=wgn(kt,1,0);%генерация массива белого Гауссова шума
% for k=1:N % генерация гармонического сигнала
%  s(k) = sin(2*pi*KP1*k/N)+ sin(2*pi*KP2*k/N);%1-й вид сигнала
%  f(k)=s(k)+ Q*noise(k); % суммирование сигнала и шума
```

```
% end
%Генерация  нестационарного сигнала
% s = 4*sin(10*time.*time./100.0);
% f = s(1:N)+ Q*noise(1:N);
% Чтение  сигнала из файла
%  filename = 'Lab_90.xls';%3-й вид сигнала
%  x = xlsread(filename);%3-й вид сигнала
filename = 'r3.csv';
x = importdata(filename);
f = x(1:N);
f(N) = f(N-1);
f = zeros(1,1+M/d);
i=1:1+M/d;
f(i)=x(i);

%Обнуление матрицы остатков
r = zeros(10,1+M/d);
r(1,:) = f;%Операция создания 1-й строки матрицы остатков r из массива f
%первая строка матрицы остатков равна самому модельному сигналу

%ПОЛУЧЕНИЕ ФУНКЦИЙ РАЗЛОЖЕНИЯ
for j = 1:10 %10 - максимальное количество функций разложения IMF
    h = r(j,:);%r(j,:)-операция выделения строки j из матрицы r
    maxLastCount = -10;
    for i = 1:N % N - максимальное количество экстремумов
         [ymax,imax,ymin,imin]  = extrema(h);%нахождение массивов
          ↪  локальных максимумов
        %и минимумов с помощью стандартной функции extrema, имеющейся в
         ↪  MATLAB
        if((length(imax) < 2) || (length(imin) < 2))
            break;
        end;
        kUp = spline(time(imax),ymax,time);%сплайн-интерполяция локальных
         ↪  максимумов
        kDown =  spline(time(imin),ymin,time);%сплайн-интерполяция
         ↪  локальных минимумов
        kMean = (kUp + kDown)/2;
        h = h - kMean;
        [one, maxCount] = size(ymax);
        if(abs(maxCount - maxLastCount) < 4)
            break;
        end;
        maxLastCount = maxCount;
    end;
    c(j,:) = h;

%Формирование искусственных экстремумов в начале и в конце массива
c(j,1)=c(j,2);%1-ый элемент массива Cj устанавливается равным 2-му
c(j,2)=c(j,3)+0.1*abs(c(j,3));%2-ой элемент массива Cj равен 3-му + 10%
 ↪  от модуля 3-го
c(j,3)=c(j,4)-0.1*abs(c(j,4));%3-ий элемент массива Cj равен 4-му минус
 ↪  10% от модуля 4-го
c(j,N)=c(j,N-1); %N-й элемент массива Cj устанавливается равным N-1-му
c(j,N-2)=c(j,N-3)+0.1*abs(c(j,N-3)); %N-2-й элемент массива Cj равен
 ↪  N-3-му +10% от модуля N-3-го
c(j,N-3)=c(j,N-4)-0.1*abs(c(j,N-4)) ; %N-3-й элемент массива Cj N-4-му
 ↪  минус 10% от модуля N-4-го

r(j + 1,:) = r(j,:) - c(j,:);
end;
```

```
%Вывод функций разложения IMF
figure
subplot (8, 1,1);
plot(c(1,:));title('IMF_1');subplot(8,1,2);
plot(c(2,:));title('IMF_2');subplot(8,1,3);
plot(c(3,:));title('IMF_3');subplot(8,1,4);
plot(c(4,:));title('IMF_4');subplot(8,1,5);
plot(c(5,:));title('IMF_5');subplot(8,1,6);
plot(c(6,:));title('IMF_6');subplot(8,1,7);
plot(c(7,:));title('IMF_7');

C1 = zeros(1,1+M/d);
%Суммирование функций разложения, содержащих только шум
for i = 1:NF
       C1=C1+c(i,:);%накопление шумовых функций IMF
end
i=1:N;
f1(i)=f(i)-C1(i);%исходный зашумленный сигнал за вычетом шумовых IMF
figure;plot(i,f(i),'r-');title('Исходный зашумленный сигнал')
figure
plot(i,f1(i));title('Отфильтрованный сигнал');
% Следующий блок программы комментируется, если зашумленные
% данные берутся из файла (в этом случаев
% чистый незашумленный сигнал s неизвестен)
%  for i=1:N
%      DZ(i)=f1(i)-s(i);%уровень зашумления в сигнале после
%      %вычитания функций разложения, содержащих только шум
%  end
%  SKO=std(DZ)%СКО полной погрешности
%  KSKO=Q/SKO %коэффициент подавления шума
pause;
close all; %закрытие всех окон графического вывода
clear; %очистка WorkSpace
```

# 7. Листинг Matlab. extrema.m

```
function [xmax,imax,xmin,imin] = extrema(x)
%EXTREMA   Gets the global extrema points from a time series.
%   [XMAX,IMAX,XMIN,IMIN] = EXTREMA(X) returns the global minima and
 ↪  maxima
%   points of the vector X ignoring NaN's, where
%    XMAX - maxima points in descending order
%    IMAX - indexes of the XMAX
%    XMIN - minima points in descending order
%    IMIN - indexes of the XMIN
%
%   DEFINITION (from http://en.wikipedia.org/wiki/Maxima_and_minima):
%   In mathematics, maxima and minima, also known as extrema, are points
 ↪  in
%   the domain of a function at which the function takes a largest value
%   (maximum) or smallest value (minimum), either within a given
%   neighbourhood (local extrema) or on the function domain in its
 ↪  entirety
%   (global extrema).
%
%   Example:
```

```
%       x = 2*pi*linspace(-1,1);
%       y = cos(x) - 0.5 + 0.5*rand(size(x)); y(40:45) = 1.85;
 ↪   y(50:53)=NaN;
%       [ymax,imax,ymin,imin] = extrema(y);
%       plot(x,y,x(imax),ymax,'g.',x(imin),ymin,'r.')
%
%   See also EXTREMA2, MAX, MIN

%   Written by
%   Lic. on Physics Carlos AdriAn Vargas Aguilera
%   Physical Oceanography MS candidate
%   UNIVERSIDAD DE GUADALAJARA
%   Mexico, 2004
%
%   nubeobscura@hotmail.com

% From        : http://www.mathworks.com/matlabcentral/fileexchange
% File ID     : 12275
% Submited at: 2006-09-14
% 2006-11-11 : English translation from spanish.
% 2006-11-17 : Accept NaN's.
% 2007-04-09 : Change name to MAXIMA, and definition added.


xmax = [];
imax = [];
xmin = [];
imin = [];

% Vector input?
Nt = numel(x);
if Nt ~= length(x)
 error('Entry must be a vector.')
end

% NaN's:
inan = find(isnan(x));
indx = 1:Nt;
if ~isempty(inan)
 indx(inan) = [];
 x(inan) = [];
 Nt = length(x);
end

% Difference between subsequent elements:
dx = diff(x);

% Is an horizontal line?
if ~any(dx)
 return
end

% Flat peaks? Put the middle element:
a = find(dx~=0);              % Indexes where x changes
lm = find(diff(a)~=1) + 1;    % Indexes where a do not changes
d = a(lm) - a(lm-1);          % Number of elements in the flat peak
a(lm) = a(lm) - floor(d/2);   % Save middle elements
a(end+1) = Nt;

% Peaks?
```

```
xa  = x(a);              % Serie without flat peaks
b = (diff(xa) > 0);      % 1  ⇒  positive slopes (minima begin)
                         % 0  ⇒  negative slopes (maxima begin)
xb  = diff(b);           % -1 ⇒  maxima indexes (but one)
                         % +1 ⇒  minima indexes (but one)
imax = find(xb == -1) + 1; % maxima indexes
imin = find(xb == +1) + 1; % minima indexes
imax = a(imax);
imin = a(imin);

nmaxi = length(imax);
nmini = length(imin);

% Maximum or minumim on a flat peak at the ends?
if (nmaxi==0) && (nmini==0)
 if x(1) > x(Nt)
  xmax = x(1);
  imax = indx(1);
  xmin = x(Nt);
  imin = indx(Nt);
 elseif x(1) < x(Nt)
  xmax = x(Nt);
  imax = indx(Nt);
  xmin = x(1);
  imin = indx(1);
 end
 return
end

% Maximum or minumim at the ends?
if (nmaxi==0)
 imax(1:2) = [1 Nt];
elseif (nmini==0)
 imin(1:2) = [1 Nt];
else
 if imax(1) < imin(1)
  imin(2:nmini+1) = imin;
  imin(1) = 1;
 else
  imax(2:nmaxi+1) = imax;
  imax(1) = 1;
 end
 if imax(end) > imin(end)
  imin(end+1) = Nt;
 else
  imax(end+1) = Nt;
 end
end
xmax = x(imax);
xmin = x(imin);

% NaN's:
if ~isempty(inan)
 imax = indx(imax);
 imin = indx(imin);
end

% Same size as x:
imax = reshape(imax,size(xmax));
imin = reshape(imin,size(xmin));
```

```
137
138 % Descending order:
139 [temp,inmax] = sort(-xmax); clear temp
140 xmax = xmax(inmax);
141 imax = imax(inmax);
142 [xmin,inmin] = sort(xmin);
143 imin = imin(inmin);
144
145
146 % Carlos AdriAn Vargas Aguilera. nubeobscura@hotmail.com
```