Цифровые фильтры

1 Введение

Цифровой фильтр - фильтр, обрабатывающий цифровой сигнал с целью выделения и (или) подавления определённых частот этого сигнала.

Разделяют два больших класса цифровых фильтров:

- Фильтры с конечной импульсной характеристикой (**КИХ-фильтры, FIR**)
- Фильтры с бесконечной импульсной характеристикой (**БИХ-фильтры, IIR**)

Зачем они нужны? Чем нас не устраивают аналговые фильтры? Давайте рассмотрим преимущества и недостатки цифровых фильтров.

Преимущества цифровых фильтров:

- Высокая точность и воспроизводимость (у аналоговых фильтров определяется разбросом номиналов компонентов)
- Гибкость (можно изменять характеристику, не затрагивая аппаратную часть)
- Компактность (например, аналоговые ФНЧ могут занимать много места, в то время как цифровые имеют одинаковые габариты при любой форме АЧХ в пределах одной частоты дискретизации)

Недостатки цифвровых фильтров:

- Сложность работы в реальном времени (все вычисления должны пройти менее, чем за период дискретизации)
- Высокая стоимость

Начнём изучение фильтров с наиболее простого типа - КИХ-фильтров.

2 КИХ-фильтры

Давайте сразу начнём с примера. Сгенерируем в Matlab зашумлённый сигнал частотой 0.5 Гц, оцифрованный с частотй дискретизации 10 Гц. Для начала, объявим переменную `fs`, которой присвоим значение частоты дискретизации, массив временных отсчётов `ts`, а переменной `N` присвоим количество получившихся отсчётов:

Листинг 1 – Скользящее среднее, часть 1

```
1 fs = 10;
2 ts = 0: 1/fs : 10-1/fs;
3 N = length(ts);
```

Теперь создадим сам синусоидальный сигнал `x` с частотой 0.5 Гц, зашумлённый случайным сигналом, амплитуда которого изменяется в диапазоне от `a=-0.01` до `b=0.1`:

Листинг 2 – Скользящее среднее, часть 2

```
4  a = -0.01;
5  b = 0.1;
6  x = (a + (b - a) * rand(1, N)).*sin(2*pi*0.5*ts);
```

Построим график сигнала `x`:

Листинг 3 – Скользящее среднее, часть 3

```
7  plot(x), grid on;
8  xlabel('Время');
9  ylabel('Амплитуда');
10 title('Сигнал x(n)');
```

Полученный результат показан на рисунке 1.

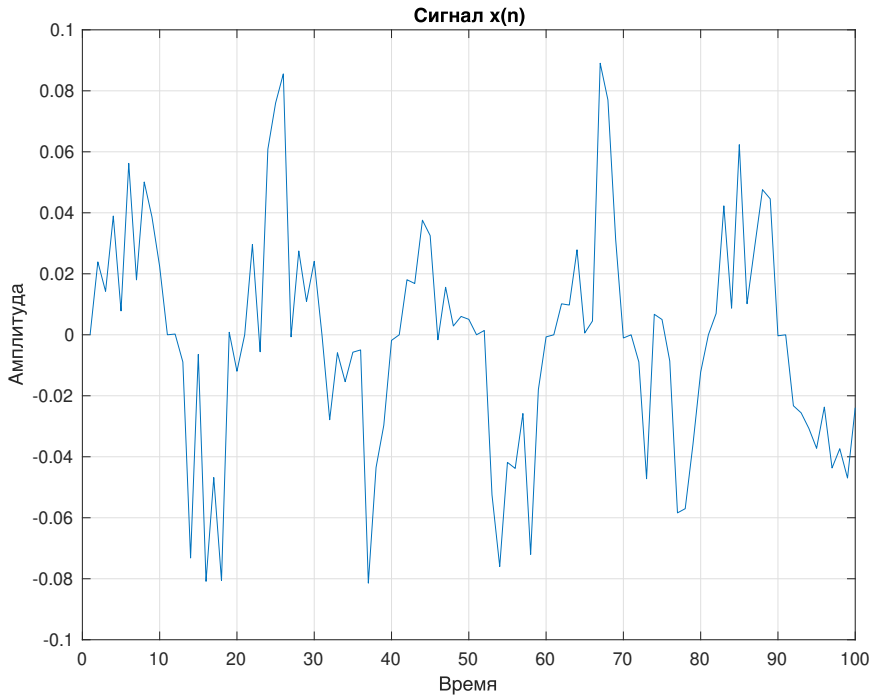

Рис. 1 – Сгенерированный сигнал `x(n)`

Тепрь возьмём первые пять отсчётов с `x(1)` по `x(5)`, найдём их среднее арифметическое, запишем его в новый массив в элемент `y(1)`, затем возьмём отсчйты от `x(2)` по `x(6)`, также рассчитаем их среднее значение и запишем его в элемент `y(2)` и так далее, пока не пройдём по всему сигналу:

Листинг 4 – Скользящее среднее, часть 4

```
11 y = zeros(1,N+6);
12 for i = 6 : length(x)
```

```
    y(i) = (x(i-1) + x(i-2) + x(i-3) + x(i-4) + x(i-5))/5;
end
```

А теперь построим сигналы `x` и `y` друг под другом в одном окне:

Листинг 5 – Скользящее среднее, часть 5

```
subplot(2,1,1);
plot(x), grid on;
xlabel('Время');
ylabel('Амплитуда');
title('Сигнал x(n)');

subplot(2,1,2);
plot(y(1:100)), grid on;
xlabel('Время');
ylabel('Амплитуда');
title('Сигнал y(n)');
```

Результаты выполнения скрипта показаны на рисунке 2.

И что же мы видим? Сигнал `y(n)` (на рисунке снизу) похож на сигнал `x(n)` (сверху), однако имеет некоторую задержку и более гладкую форму. Получается, что мы только что применили к сигналу `x(n)` фильтр нижних частот (ФНЧ)! Спроектированный нами фильтр назвается **скользящее среднее**. Уравнение данного фильтра имеет вид:

$$y[n] = \sum_{k=0}^{M-1} h(k) \cdot x(n-k), \text{ где} \sum_{k=0}^{M} h(k) = 1 \tag{1}$$

Структурная схема данного фильтра представлена на рисунке 3.

В каждый момент времени берётся текущий отсчёт и 4 предыдущих, каждый из которых домножается на коэффициент $1/5$, затем эти результаты складываются и полученная сумма поступает на выход. Процедура повторяется, пока весь сигнал не будет обработан.

2.1 Свёртка

А что, если вместо коэффициентов $1/5$ взять что-то другое? На самом деле, уравнение (1) представляет собой частный случай операции **свёртки**:

$$y[n] = x[n] * h[n] = \sum_{k=0}^{M-1} h(k) \cdot x(n-k), \tag{2}$$

где $M = N_1 + N_2 - 1$, N_1 – длина $x[n]$, N_2 – длина $h[n]$

Это уравнение и является уравнением КИХ-фильтра. Его структурная схема представлена на рисунке 4.

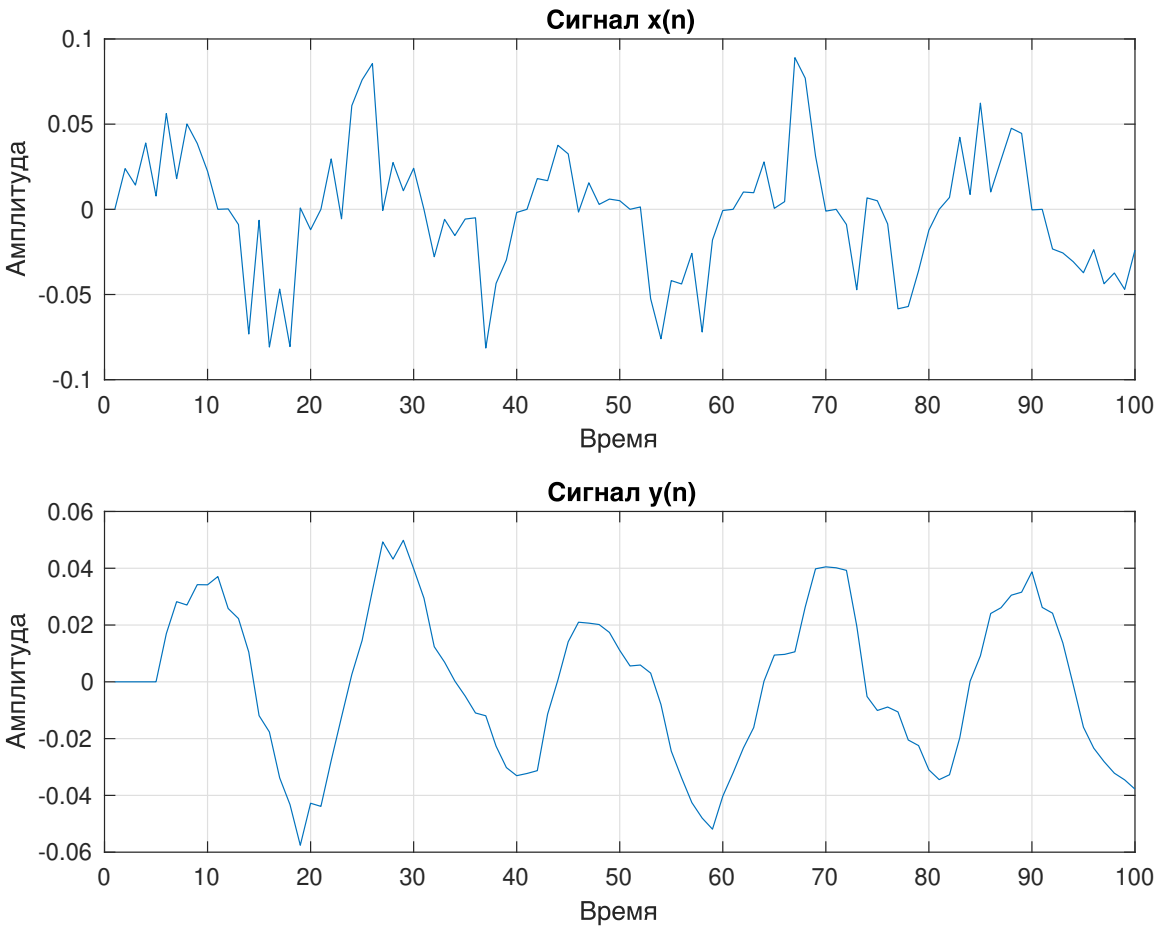

Рис. 2 – Результаты работы фильтра "скользящее среднее"

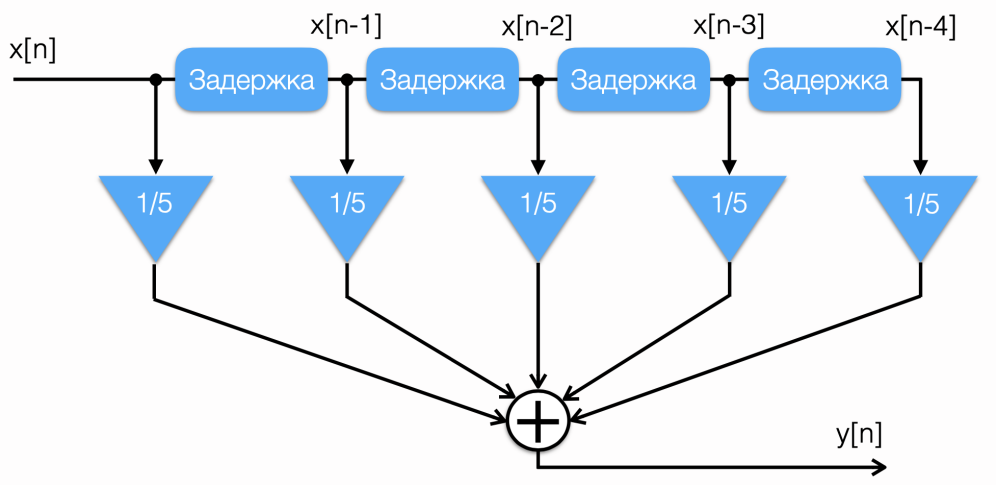

Рис. 3 – Структурная схема фильтра "скользящее среднее состоящего из пяти отсчётов

2.2 Теорема о свёртке

Теорема о свёртке звучит следующим образом:

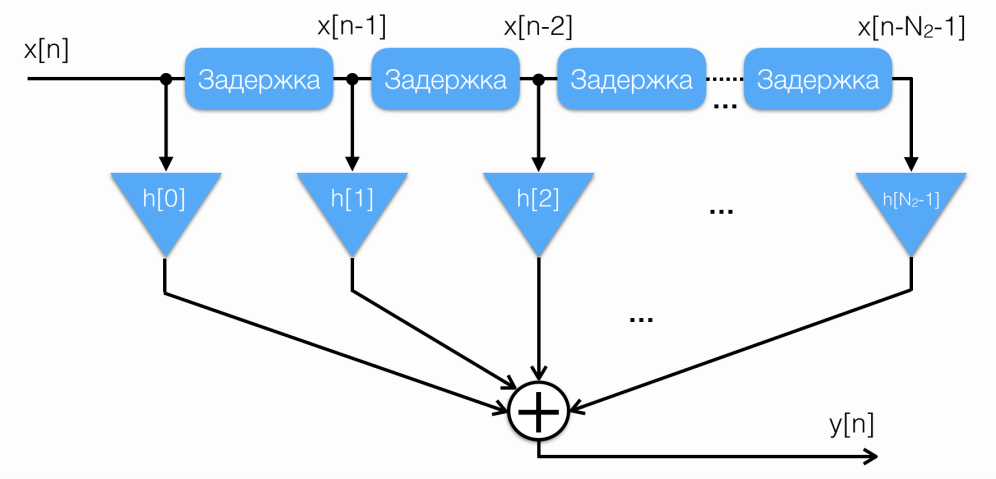

Рис. 4 – Структурная схема КИХ-фильтра

Теорема 1 *Если две последовательности во временной области $x[n]$ и $h[n]$ имеют дискретные преобразования Фурье (ДПФ) $X[m]$ и $H[m]$ соответственно, то ДПФ свёртки $x[n] * h[n]$ представляет собой произведение $X[m] \cdot H[m]$:*

$$x[n] * h[n] \xleftrightarrow[ОДПФ]{ДПФ} X[m] \cdot H[m] \tag{3}$$

И действует это в обе стороны. Получается, что, когда мы делаем свёртку нашего исходного сигнала с коэффициентами фильтра, мы перемножаем амплитудно-частотную характеристику (АЧХ) исходного сигнала и АЧХ фильтра, тем самым убираем ненужные частоты. А это и есть фильтрация.

Мы выше говорили: "фильтры с конечной импульсной характеристикой, фильтры с бесконечной импульсной характеристикой". А что же такое "импульсная характеристика"? Давайте вспомним на примере фильтров.

2.3 Импульсная характеристика КИХ-фильтра

Импульсная характеристика фильтра - это выходная последовательность фильтра во временной области при подаче на вход фильтра **дельта-функции Дирака** - единственного отсчёта, равного единице (единичного импульса), которому предшествуют и за которым следуют нулевые отсчёты.

Возьмём предыдущий пример из Matlab и вместо исходного сигнала `x` подставим в него функцию Дирака:

Листинг 6 – Импульсная характеристика скользящего среднего

```
clear;
fs = 10;
ts = 0: 1/fs : 10-1/fs;
N = length(ts);

x = zeros(N,1);
x(5) = 1;
```

```
8
9  subplot(2,1,1);
10 stem(x), grid on;
11 xlabel('Время');
12 ylabel('Амплитуда');
13 title('Функция Дирака');
14
15 y = zeros(1,N+6);
16 for i = 6 : length(x)
17    y(i) = (x(i-1) + x(i-2) + x(i-3) + x(i-4) + x(i-5))*1/5;
18 end
19
20 subplot(2,1,2);
21 stem(y(1:100)), grid on;
22 xlabel('Время');
23 ylabel('Амплитуда');
24 title('Импульсная характеристика');
```

Результат выполнения скрипта показан на рисунке 5.

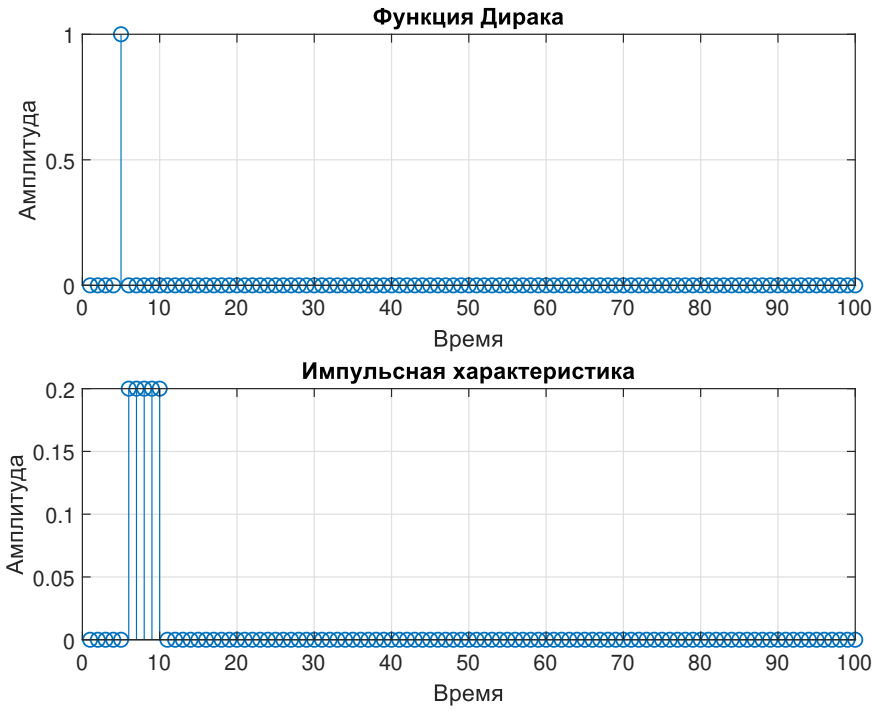

Рис. 5 – Функция Дирака и реакция на неё скользящего среднего из пяти отсчётов

Из рисунка видно, что импульсная характеристика фильтра представляет собой 5 отсчётов амплитудой

0.2 (или 1/5). Получается, что мы с вами видим ни что иное, как коэффициенты фильтра. Поэтому **коэффициенты КИХ-фильтра также называют его импульсной характеристикой**.

2.4 Проектирование КИХ-фильтров

Теперь возникает вопрос: как рассчитать коэффициенты, чтобы получить требуемую АЧХ фильтра? Самый простой способ: "рисуем"необходимую АЧХ в частотной области, делаем обратное ДПФ от этой АЧХ и получаем набор коэффициентов во временной области. Наверняка, каждый из вас мечтает увидеть фильтр с идеально прямоугольной АЧХ. Возможно ли такое? Давайте разбираться. Что из себя представляет ДПФ от идеально прямоугольного сигнала? Правильно, функцию *sinc*, которую мы рассматривали, когда изучали растекание спектра ДПФ (см. рисунок 6).

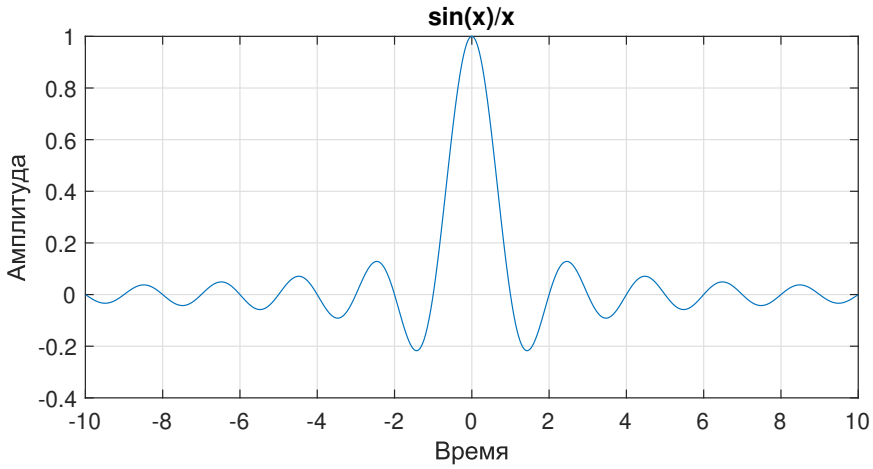

Рис. 6 – Функция `sinc`

Данная функция бесконечна во временной области, поэтому для реализации идеально прямоугольной АЧХ нам потребуется бесконечное количество коэффициентов фильтра. Получается, чем больше мы используем коэффициентов `h[n]`, тем больше АЧХ фильтра будет похожа на прямоугольную. Вроде бы логично – возьми побольше коэффициентов, получишь хороший фильтр. Но не всё так просто: в реальности мы не можем увеличивать количество коэффициентов фильтра, т.к. это приводит к дополнительным операциям перемножения, каждая из которых вызывает задержку. Для того, чтобы система работала в реальном времени, нужно, чтобы все вычисления по формуле (2) производились за время, не превышающее один период дискретизации. Вот и простейшее ограничение.

Давайте попробуем на примере. Создадим сигнал `x(n)`, состоящий из `N` нулевых отсчётов, затем присвоим первым 200 отсчтётам значение 1. Это будет наша требуемая форма АЧХ:

Листинг 7 – Анализ АЧХ фильтра, часть 1

```
clear;
fs = 1000;            % частота дискретизации
ts = 0:1/fs:1-1/fs;   % массив временных отсчётов
N = length(ts);       % длина массива временных отсчётов
```

```
5
6  x = zeros(N,1);
7  x(1:200) = 1;
```

Рассчитаем ОДПФ от сигнала `x(n)`, для удобства сдвинем его с помощью `ifftshift`, отбросим мнимую часть и присвоим полученный результат сигналу `y(n)`. Далее построим графики сигнала `x(n)`, и, чтобы лучше рассмотреть сигнал `y(n)`, график его 200 центральных отсчётов.

Листинг 8 – Анализ АЧХ фильтра, часть 2

```
8  y = real(ifftshift(ifft(x)));
9
10 subplot(2,1,1);
11 plot(x), grid on;
12 xlabel('Частотные отсчёты');
13 ylabel('Амплитуда');
14 title('Ожидаемая АЧХ, x(n)');
15
16 subplot(2,1,2);
17 plot(N/2-100:N/2+100-1,y(N/2-100:N/2+100-1)), grid on;
18 xlabel('Временные отсчёты');
19 ylabel('Амплитуда');
20 title('Фрагмент полученной импульсной характеристики, y(n)');
```

Результат выполнения скрипта показан на рисунке 7.

Теперь давйте анализировать реальную АЧХ от спроектированного фильтра. Для этого создадим сигнал `a(n)`, состоящий из `N` нулевых отсчётов, затем присвоим первому отсчёту 1. Таким образом, получим функцию Дирака, спектр которой является константой на всей частотной оси:

Листинг 9 – Анализ АЧХ фильтра, часть 3

```
21 a = zeros(N,1);
22 a(1) = 1;
```

Теперь попробуем пропустить функцию Дирака через спроектированный ранее фильтр `y(n)`, при этом будем использовать разное количество отсчётов `y(n)`:

Листинг 10 – Анализ АЧХ фильтра, часть 4

```
23 figure;
24
25 Nf1 = 5;
26 af1 = filter(y(N/2-Nf1:N/2+Nf1-1),1,a);
```

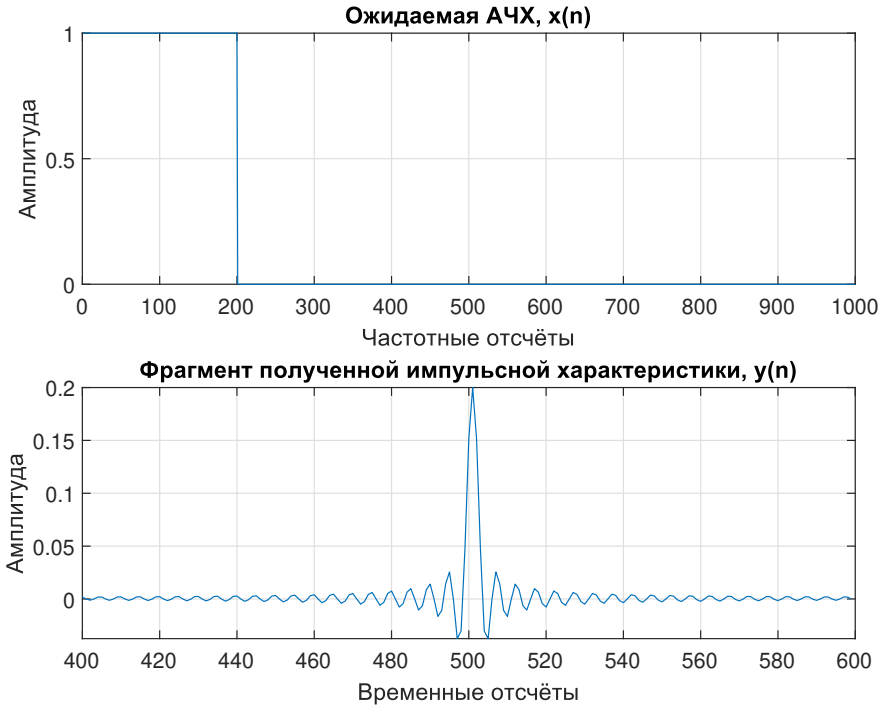

Рис. 7 – Требуемая форма АЧХ фильтра `x(n)`
и 200 центральных отсчётов его импульсной характеристики `y(n)`

```
27 subplot(2,2,1);
28 plot(abs(fft(af1))), grid on;
29 xlabel('Частотные отсчёты');
30 ylabel('Амплитуда');
31 title('АЧХ фильтра, N=10');
32
33 Nf2 = 10;
34 af2 = filter(y(N/2-Nf2:N/2+Nf2-1),1,a);
35 subplot(2,2,2);
36 plot(abs(fft(af2))), grid on;
37 xlabel('Частотные отсчёты');
38 ylabel('Амплитуда');
39 title('АЧХ фильтра, N=20');
40
41 Nf3 = 25;
42 af3 = filter(y(N/2-Nf3:N/2+Nf3-1),1,a);
43 subplot(2,2,3);
44 plot(abs(fft(af3))), grid on;
```

```
xlabel('Частотные отсчёты');
ylabel('Амплитуда');
title('АЧХ фильтра, N=50');

Nf4 = 50;
af4 = filter(y(N/2-Nf4:N/2+Nf4-1),1,a);
subplot(2,2,4);
plot(abs(fft(af4))), grid on;
xlabel('Частотные отсчёты');
ylabel('Амплитуда');
title('АЧХ фильтра, N=100');
```

Результат выполнения скрипта показан на рисунке 8.

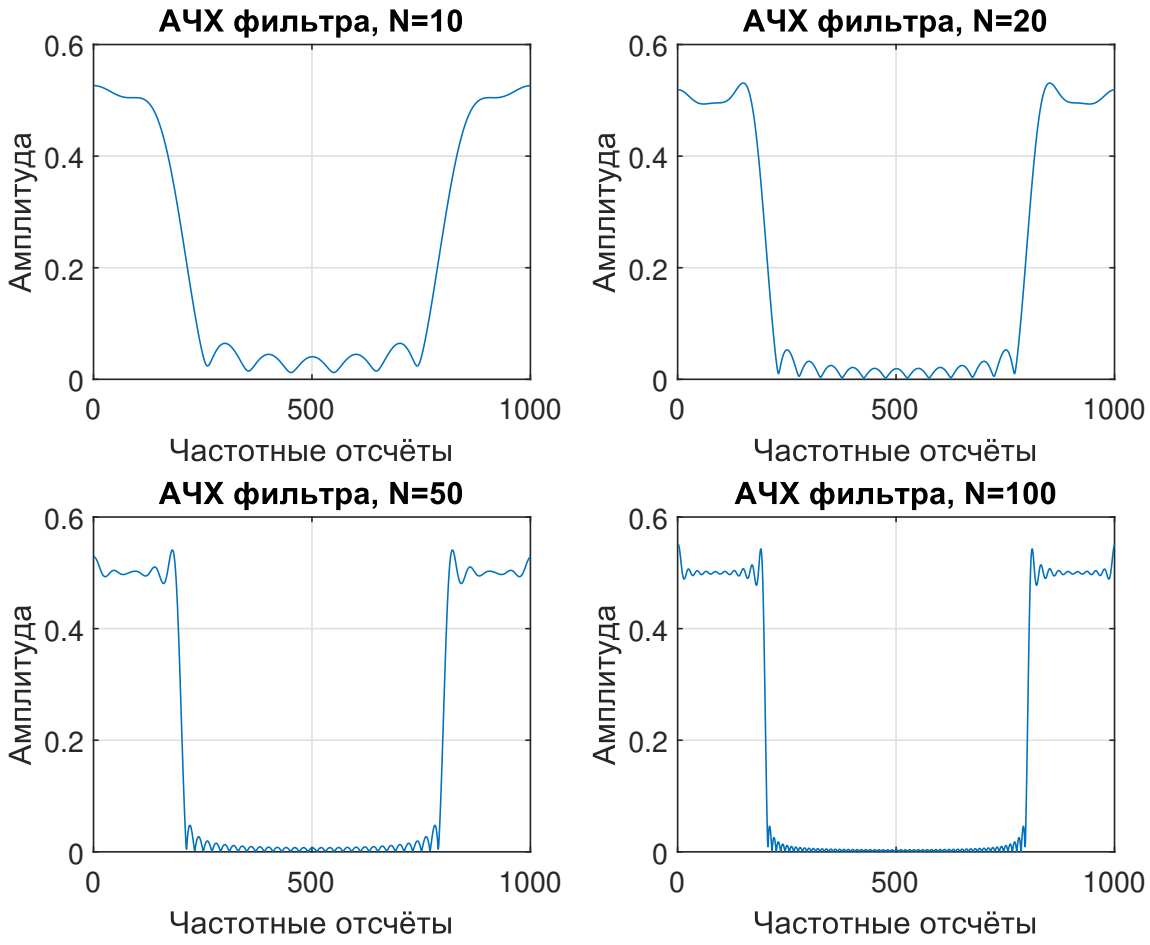

Рис. 8 – Форма АЧХ фильтра в зависимости от количества отсчётов

Из графиков видно, что, чем больше мы используем отсчётов нашего фильтра, тем более его характеристика становится похожа на идеальную. Однако, стоит обратить внемание, что на вершине всех наших фильтров

видны пульсации АЧХ независимо от того, сколько используется отсчётов в импульсной характеристике. Эти пульсации называются **пульсации Гиббса** и возникают из-за медленной сходимости ряда Фурье, которая обусловлена наличием разрыва функции на частоте среза полосы пропускания фильтра. С увеличением числа отсчетов уменьшается длительность выбросов на вершине АЧХ, но их амплитуда не меняется и составляет примерно 9% от амплитуды АЧХ на частоте среза.

2.5 Окна при проетировании КИХ-фильтров:

Запишем выражение (3) следующим образом:

$$h^{\infty}[k] \cdot w[k] \xleftrightarrow[\text{ОДПФ}]{\text{ДПФ}} H^{\infty}[m] * W[m] \tag{4}$$

где:

- $h^{\infty}[k]$ – бесконечная импульсная характеристика идеального фильтра
- $w[k]$ – окно, наложенное на бесконечную импульсную характеристику
- $H^{\infty}[m]$ и $W[m]$ – их ДПФ

Получается, что, когда мы берём какое-то ограниченное количество отсчётов от импульсной характеристики идеального фильтра, мы домножаем эту импульсную характеристику на прямоугольное окно, спектр которого нам известен (рисунок 6). Эта процедура во временной области эквивалентна свёртке спектра идеальной характеристики и спектра ограниченной импульсной характеристики в частотной области. Ниже приведён листинг, моделирующий выражение (4):

Листинг 11 – Моделирование выражения (4)

```
clear;
fs = 100;                 % частота дискретизации
ts = -20:1/fs:20-1/fs;    % временные отсчёты
N = length(ts);           % длина массива временных отсчётов

% Идеальный прямоугольный спектр от бесконечного числа отсчётов
Hinf = zeros(N,1);
Hinf(N/2-750:N/2+749) = 1;

% Спектр прямоугольного окна - sinc(x)
W = sinc(ts);

subplot(2,1,1);
plot(Hinf), grid on, hold on;
plot(W);
```

```
16 xlabel('Частотные отсчёты');
17 ylabel('Амплитуда');
18 title('АЧХ фильтра с бесконечной характеристикой Hinf(n) и окна W(n)');
19
20 % Свёртка спектров
21 c = conv(Hinf, W);
22
23 subplot(2,1,2);
24 plot(c), grid on;
25 xlabel('Частотные отсчёты');
26 ylabel('Амплитуда');
27 title('Свёртка Hinf(n) и W(n)');
```

Результат выполнения скрипта показан на рисунке 9.

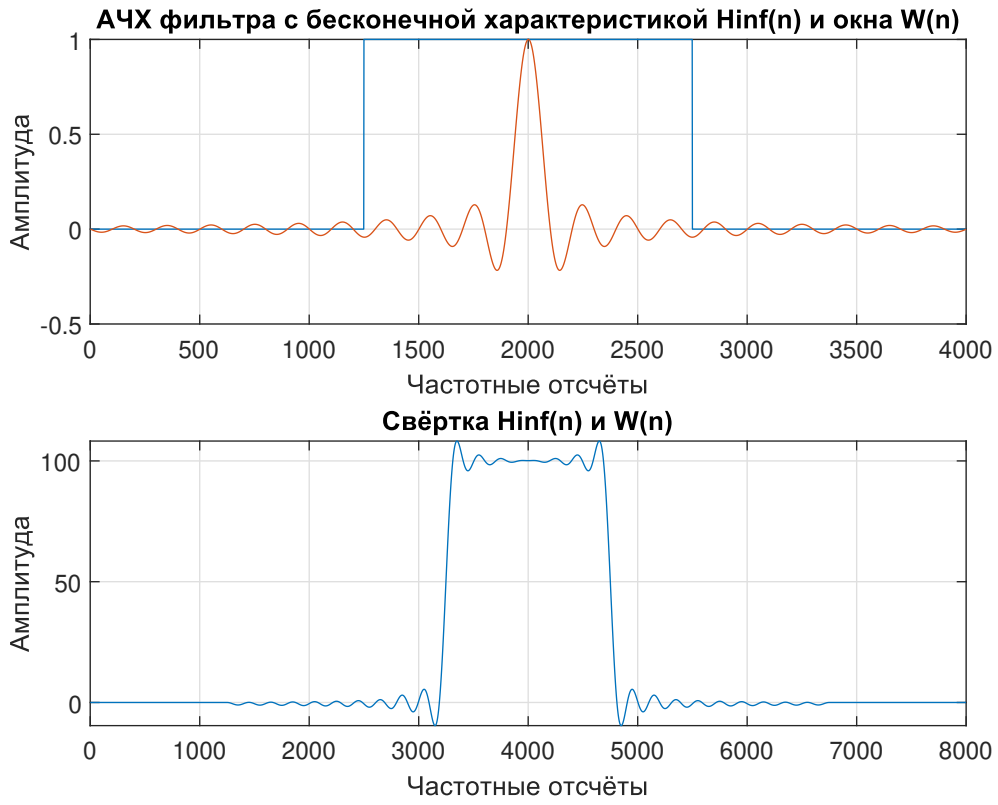

Рис. 9 – Форма АЧХ фильтра, взвешенного окном, в зависимости от количества отсчётов

Чтобы уменьшить амплитуду пульсаций, как и в случае с ДПФ, при проектировании КИХ-фильтров используют окна, отличные от прямоугольного. Давайте модифицируем наш листинг 7 и добавим в строчках 26, 34, 42, 50 домножение на окно Хэмминга:

Листинг 12 – Модификация листинга 10

```
...
af1 = filter(y(N/2-Nf1:N/2+Nf1-1).*hamming(Nf1*2),1,a);
...
af2 = filter(y(N/2-Nf2:N/2+Nf2-1).*hamming(Nf2*2),1,a);
...
af3 = filter(y(N/2-Nf3:N/2+Nf3-1).*hamming(Nf3*2),1,a);
...
af4 = filter(y(N/2-Nf4:N/2+Nf4-1).*hamming(Nf4*2),1,a);
...
```

Полученные результаты показаны на рисунке 10.

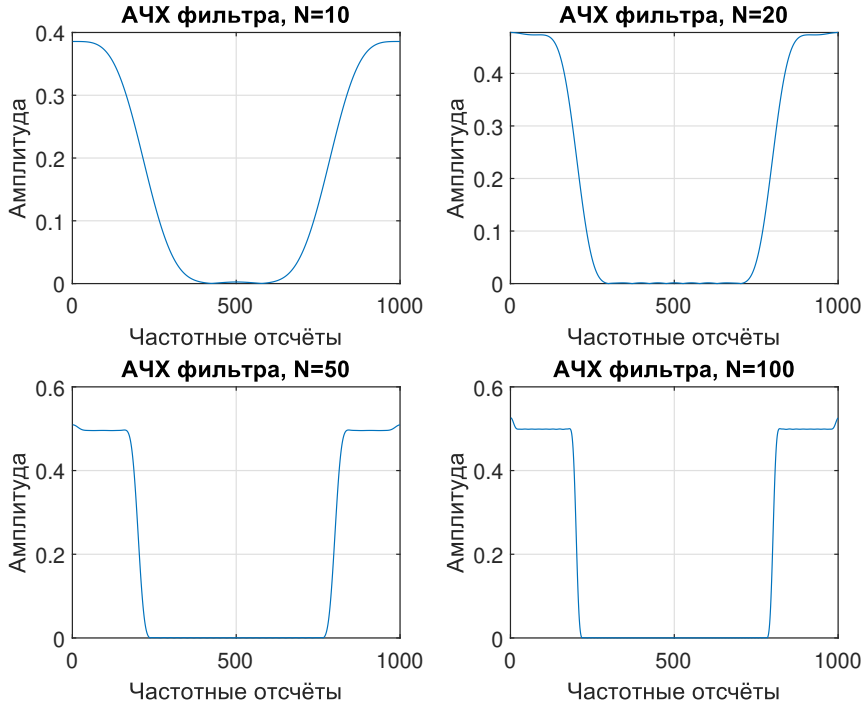

Рис. 10 – Форма АЧХ фильтра, взвешенного окном, в зависимости от количества отсчётов

Другое дело! Пульсации ушли, однако АЧХ фильтра стала более "заваленной".

2.6 Задания к семинару:

1. Разработать свой усредняющий фильтр для произвольного сигнала и построить его импульсную характеристику.

2. Изучить пакет Matlab filterDesigner.

3. Сгенерировать сигнал, содержащий несколько частотных составляющих, затем разработать с помощью filterDesigner КИХ-фильтр, который убирает и оставляет как минимум одну из этих частотных составляющих.

4. Разработать и применить КИХ-фильтр произвольной АЧХ к произвольному звуковому файлу.

5. Разработать в пакете Matlab Simulink эквалайзер на базе КИХ-фильтров. Количество каналов должно быть не менее четырёх.