1. Цифровой фильтр задан разностным уравнением (БИХ-фильтр (IIR)):

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k)$$

$$y(k) = 0.0089 \cdot x(k) - 0.0045 \cdot x(k-1) - 0.0045 \cdot x(k-2) + 0.0089 \cdot x(k-3) +$$
$$+ 2.5641 \cdot y(k-1) - 2.2185 \cdot y(k-2) + 0.6456 \cdot y(k-3)$$

Разностное уравнение задает зависимость фильтра от входного сигнала и прошлых значений. Уравнение описывает поведение фильтра во временной области.

У КИХ-фильтра:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$
$$H(z) = \sum_{k=0}^{N-1} h(k)z^{-k}$$

Передаточная характеристика БИХ-фильтра:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

Это математическая модель фильтра, позволяет понять, как он изменяет сигнал на всех частотах. Представляет собой полное описание фильтра в частотной области.

Для перехода к АЧХ в передаточную характеристику подставляем $z = e^{j\omega}$. Она показывает, какой коэффициент усиления/ослабления имеет сигнал на каждой частоте. Показывает, какие частоты фильтр пропускает, какие подавляет.

Получить фазо-частотную характеристику:

$$H(e^{j\omega}) = a + jb$$

$$\phi(\omega) = \arg(H(e^{j\omega})) = \arctan 2(b, a)$$

Она показывает, насколько «сдвигается» фаза каждого синусоидального компонента при прохождении через фильтр.

Групповая задержка:

$$\tau_g(\omega) = -\frac{d}{d\omega} \arg(H(e^{j\omega}))$$

Показывает, насколько задерживается информация на каждой частоте. Идеальный фильтр имеет постоянную групповую задержку.

В matlab:

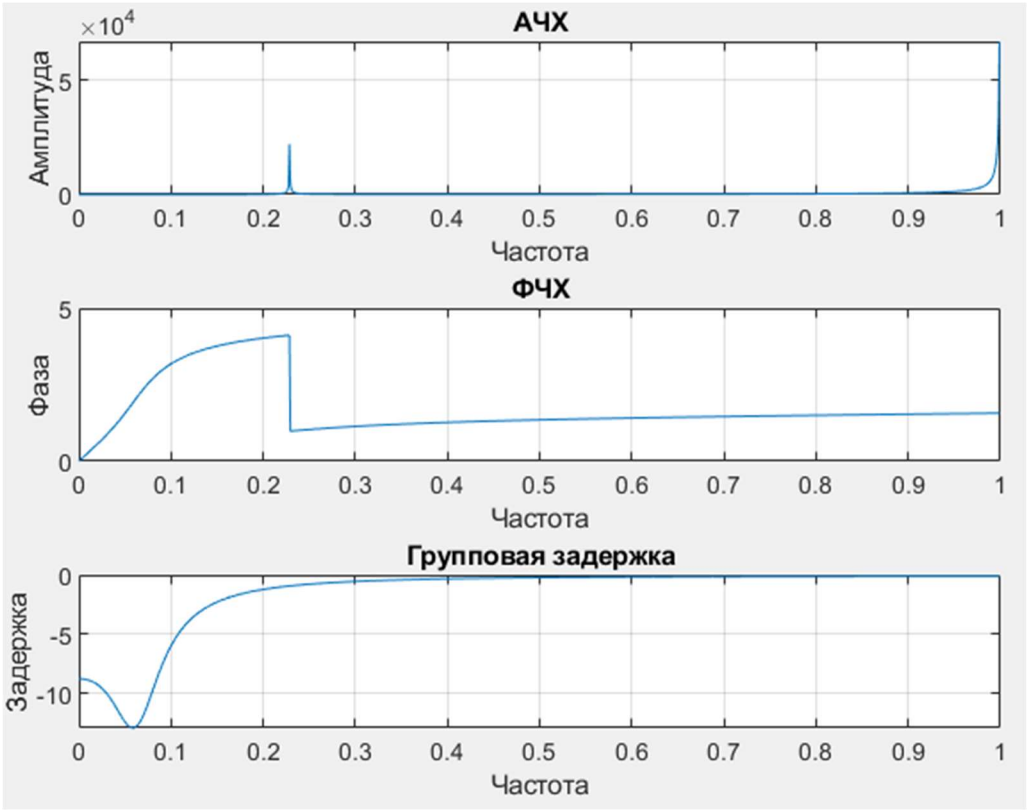
```
a = [0.0089, -0.0045, -0.0045, 0.0089];
b = [1, -2.5641, 2.2185, -0.6456];

[H, w] = freqz(b, a, 1024); // частотная характеристика фильтра, H – массив комплексных значений на 1024 точках, w – соответствующие частоты в диапазоне от 0 до pi.

figure;
subplot(3,1,1);
plot(w/pi, abs(H)); // abs() – модуль комплексного числа
title("АЧХ");
xlabel("Частота");
ylabel("Амплитуда");
grid on;

subplot(3,1,2);
plot(w/pi, unwrap(angle(H))); // angle() – фаза комплексного значения, unwrap() --устраняет скачки +-pi
title("ФЧХ");
xlabel("Частота");
ylabel("Фаза");
grid on;

[gd, w_gd] = grpdelay(b,a,1024); // gd – массив значений групповой задержки, w_gd – частоты, соответствующие каждой задержке
subplot(3,1,3);
plot(w_gd/pi, gd);
title("Групповая задержка");
xlabel("Частота");
ylabel("Задержка");
grid on;
```

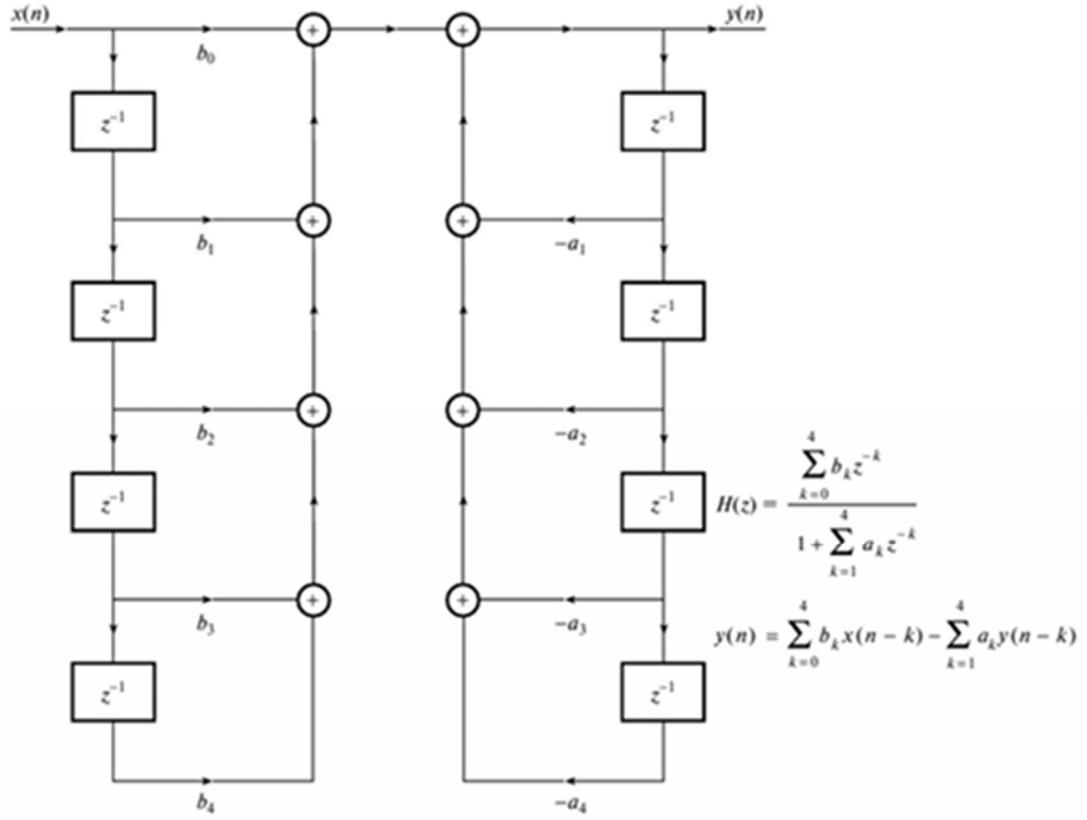


Из АЧХ видно, что везде кроме одной частоты (~0.22) почти нулевая амплитуда. Отсюда можно сделать вывод, что это полосовой фильтра (полосовой резонатор, пропускающий только очень узкий диапазон частот, подавляя все остальные).

По ФЧХ видно, что фаза плавно возрастает до ~0.22, затем резко падает вниз, потом снова плавно растет. Фильтр имеет фазовую неоднородность, это может искажать форму сигнала.

Групповая задержка имеет провал до ~0.22, затем становится положительной и стабильной. Это значит, что сигнал сильно искажается в районе этих частота. Такие значения появляются в области фазовых скачков.

Исходя из разностного уравнения можно сделать вывод, что фильтру соответствует прямая структура:



Уровень подавления фильтра в полосе заграждения – насколько сильно фильтр ослабляет сигналы на частотах, которые он не должен пропускать.

Для амплитуды A на какой-то частоте:

$$\text{Затухание (подавление)} = 20\log_{10}(A)$$

Для нахождения уровня подавления сначала переводим амплитуду в логарифмическую шкалу (дБ):

$$\text{Уровень в дБ} = 20\log_{10}(|H|)$$

Затем находим максимальное значение в области подавления. Максимальное, потому что это область подавления и нас интересует наихудший случай.

В matlab:

```
H_db = 20*log10(abs(H));
stopband_indicies = find((w/pi < 0.21 | w/pi > 0.24) & w/pi < 0.9);
max_H_in_stopband = max(H_db(stopband_indicies));
fprintf("Уровень подавления в полосе заграждения: %.2f дБ\n",
max_H_in_stopband);
```

- Фильтр не является FIR-фильтром. Это IIR-фильтр.
- Полосовой фильтр
- 3 порядок фильтра
- Уровень подавления: 56.24 дБ

2. На вход системы поступают данные с частотой сэмплирования Fs. Как увеличить частоту сэмплирования в 3 раза/уменьшить в 5/3 раза с минимальным изменением свойств сигнала:

Способы изменения частоты сэмплирования:
1. Интерполяция и децимация через вставку нулей и фильтрацию. Хорошо подходит для изменения на целое число или на рациональную дробь.

Алгоритм для повышения частоты (интерполяция):
- вставить L-1 нулей между каждым отсчётом (L – коэффициент интерполяции)
- пропустить сигнал через низкочастотный фильтр с полосой [0, Fs/(2L)]

```
y1 = randn(1, 333);
L = 3;

fc = 1/L;
N = 3;
h = fir1(N, fc, hamming(N+1));

y1_upsampled = zeros(1, length(y1)*L); // Создает новый вектор в L
раз больше исходного
y1_upsampled(1:L:end) = y1; // Вставляет значения y1 через каждый L
позиций

y2 = conv(y1_upsampled, h, "same");

figure;
subplot(2,1,1);
x1 = 0:(length(y1) - 1);
plot(x1, y1);
grid on;

subplot(2,1,2);
x2 = 0:(length(y2) - 1);
plot(x2, y2);
grid on;
```

Алгоритм для понижения частоты (децимации):

- пропустить сигнал через низкочастотный фильтр с полосой [0, Fs/(2M)] (M – коэффициент децимации)
- отобрать каждый M-й отсчёт

```
y1 = randn(1, 1000);
M = 3;

fc = 1/(2 * M);
N = 3;
h = fir1(N, fc, hamming(N+1)); // Проектирование КИХ-фильтра с
частотой среза и окном Хэмминга. fir1() – генерация фильтров на
основе sinc + окна

y1_filtered = conv(y1, h, "same");

y2 = y1_filtered(1:M:end);

figure;
subplot(2,1,1);
x1 = 0:(length(y1) - 1);
plot(x1, y1);
grid on;

subplot(2,1,2);
x2 = 0:(length(y2) - 1);
plot(x2, y2);
grid on;
```

Комбинирования способов:

- вставка нулей
- фильтрация
- выборка каждого M-го

```
y1 = randn(1, 500);
L = 3;
M = 5;

fc = 1/(2*max(L,M));
N = 3;
h = fir1(N, fc, hamming(N+1));

y1_upsampled = zeros(1, length(y1)*L);
y1_upsampled(1:L:end) = y1;

y1_interp = conv(y1_upsampled, h, "same"); // Интерполяция
y2 = y1_filtered(1:M:end);                 // Децимация

figure;
subplot(2,1,1);
x1 = 0:(length(y1) - 1);
plot(x1, y1);
grid on;
```

```
subplot(2,1,2);
x2 = 0:(length(y2) - 1);
plot(x2, y2);
grid on;
```

Преимущества: хорошее качество, прост в реализации

Недостатки: требует вычислений (для фильтрации)

2. Полиномиальная интерполяция. Применима при нецелых коэффициентах, временном масштабировании сигналов.

Алгоритм:

Для нового момента времени вычисляется значение по известным точка с помощью:

- линейной интерполяции (по двум ближайшим точкам по бокам)

$$x(t') = x(t_1) + \frac{t' - t_1}{t_2 - t_1} \cdot (x(t_2) - x(t_1))$$

Увеличение:

```
x = rand(1,300);
t = 0:(length(x) - 1);
K = 4;

t_interp = linspace(t(1), t(end), length(t)*K - (K-1));
x_interp = zeros(1, length(t_interp));

for i = 1:length(t_interp)
    ti = t_interp(i);
    idx = find(t <= ti, 1, 'last');

    if idx == length(x)
        x_interp(i) = x(end);
    else
        t1 = t(idx);
        t2 = t(idx+1);
        y1 = x(idx);
        y2 = x(idx+1);
        x_interp(i) = y1 + (y2 - y1) * (ti - t1) / (t2 - t1);
    end
end

figure;
subplot(2,1,1);
plot(t, x);
grid on;

subplot(2,1,2);
plot(t_interp, x_interp);
grid on;
```

Уменьшение:

Просто выбор каждого K-того сигнала

Комбинирование: сначала интерполяция, потом децимация

- кубической интерполяции (по четырём точкам)
- сплайнов

Преимущества: простота, работает с произвольными частотами
Недостатки: меньшая точность и качество, может создавать искажения

3. Sinc-интерполяция. При преобразовании между сильно разными отсчётами.

Алгоритм:
Каждый новый отсчёт – сумма всех оригинальных, умноженных на sinc-функции, центрированные в этих отсчётах.

$$x(t) = \sum_n x[n] \cdot \mathrm{sinc}\left(\frac{t - nT}{T}\right), \qquad \mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

```
x = rand(1,200);
Fs = 1;
T = 1/Fs;
t_original = 0:T:(length(x)-1)*T;

L = 2;
Fs_new = Fs * L; // Или Fs-new = Fs / L; для децимации
T_new = 1/Fs_new;
t_new = 0:T_new:(length(x)-1)*T;

x_new = zeros(size(t_new));
for i = 1:length(t_new)
    ti = t_new(i);
    for n = 1:length(x)
        x_new(i) = x_new(i) + x(n) * sinc((ti - t_original(n))/T);
    end
end

figure;
subplot(2,1,1);
plot(t_original, x);
grid on;

subplot(2,1,2);
plot(t_new, x_new);
grid on;
```

Преимущества: максимальная точность
Недостатки: бесконечная длительность sinc-функции (нужна аппроксимация), большие вычислительные затраты

4. Частотное преобразование (через БПФ). Используется, когда весь сигнал известен заранее.

Алгоритм:
- БПФ сигнала

- добавление/удаление коэффициентов в нужное число раз
- обратное БПФ

Увеличение:

```
x = rand(1,200);
N = length(x);

L = 2;
new_N = N * L;

Xf = fft(x);

if mod(N,2) == 0
    Xf_new = [Xf(1:N/2), zeros(1, new_N - N), Xf(N/2+1:end)];
else
    Xf_new = [Xf(1:ceil(N/2)), zeros(1, new_N - N),
Xf(ceil(N/2)+1:end)];
end

x_new = real(ifft(Xf_new)) * L;

t_original = 0:N-1;
t_new = 0:N*L-1;

figure;
subplot(2,1,1);
plot(t_original, x);
grid on;
subplot(2,1,2);
plot(t_new, x_new);
grid on;
```

Уменьшение:

```
x = rand(1,500);
N = length(x);

L = 2;
new_N = floor(N / L);

Xf = fft(x);

if mod(N,2) == 0
    Xf_new = [Xf(1:new_N/2), Xf(end - new_N/2+1:end)];
else
    Xf_new = [Xf(1:ceil(new_N/2)), Xf(end - floor(new_N/2)+1:end)];
end

x_new = real(ifft(Xf_new)) * L;

t_original = 0:N-1;
t_new = 0:new_N-1;

figure;
subplot(2,1,1);
plot(t_original, x);
```

```
grid on;
subplot(2,1,2);
plot(t_new, x_new);
grid on;
```

Преимущества: подходит для произвольных коэффициентов, высокая точность
Недостатки: не работает в реальном времени, требуется весь сигнал заранее