МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Цифровая обработка сигналов»

Тема: Частотный анализ полиномиальных приближений

Студент гр.8382	_________________	Нечепуренко Н.А.
Студент гр.8382	_________________	Терехов А.Е.
Преподаватель	_________________	Сучков А.И.

Санкт-Петербург

2021

Цели работы

Анализ частотных характеристик известных формул полиномиального сглаживания временных рядов и формул численного интегрирования.

Основные теоретические положения

В качестве временного ряда рассматривается дискретный сигнал с шагом дискретизации, равным единице.

Под полиномиальным сглаживанием понимается аппроксимация в смысле МНК значений конечного (нечетного) числа элементов сглаживаемого ряда полиномом заданного порядка с присвоением среднему из этих элементов значения сглаживающего полинома в центре выбранного временного отрезка. Такой подход соответствует так называемому сглаживанию в скользящем окне.

В качестве исследуемых формул численного интегрирования используются квадратурные формулы Ньютона-Котеса.

Постановка задачи

Получить формулы для передаточных функций нерекурсивных фильтров, соответствующих полиномиальному сглаживанию дискретного сигнала для полиномов различного порядка и построить графики $\tilde{H}(f)$. Проинтерпретировать частотные свойства передаточных функций. Провести сопоставительный анализ частотных характеристик передаточных функций для различных степеней полиномов. Получить формулы для передаточных функций рекурсивных фильтров, соответствующих квадратурным формулам Ньютона-Котеса различного порядка. Проинтерпретировать частотные свойства передаточных функций. Провести сопоставительный анализ частотных характеристик передаточных функций для различных квадратурных формул.

Порядок выполнения работы

1. Вывести формулы для передаточной функции нерекурсивного фильтра, соответствующего сглаживанию прямой линией по 3, 5, 7 и 9 точкам. Построить графики $\tilde{H}(f)$. Проинтерпретировать частотные свойства передаточных функций для различного количества точек.
2. Вывести формулы для передаточной функции нерекурсивного фильтра, соответствующего сглаживанию полиномом второй степени по 7, 9, 11 и 13 точкам. Построить графики $\tilde{H}(f)$. Проинтерпретировать частотные свойства передаточных функций для различного количества точек.
3. Вывести формулы для передаточной функции нерекурсивного фильтра, соответствующего сглаживанию полиномом четвёртой степени по 9, 11, 13 и 15 точкам. Построить графики $\tilde{H}(f)$. Проинтерпретировать частотные свойства передаточных функций для различного количества точек.
4. Вывести формулы для передаточной функции нерекурсивного фильтра, соответствующего сглаживанию по формулам Спенсера по 13, 15 и 21 точкам. Построить графики $\tilde{H}(f)$. Проинтерпретировать частотные свойства передаточных функций для различного количества точек.
5. Построить графики из предыдущих пунктов в логарифмической шкале (Дб). Объясните, чем отличаются данные графики от полученных ранее и объясните их смысл.
6. Провести сопоставительный анализ свойств передаточных функций, полученных при выполнении пп. 1-4.
7. Вывести формулы передаточных функций рекурсивных фильтров, соответствующих квадратурным формулам прямоугольников, трапеций и Симпсона. Построить графики передаточных функций и графики отно-

шения вычисляемого в результате фильтрации значения к истинному. Проинтерпретировать частотные свойства полученных передаточных функций.

8. Вывести формулу передаточной функции рекурсивного фильтра, соответствующего квадратурной формуле:

$$y_{n+2} = y_{n-1} + \frac{1}{8}(x_{n+2} + 3x_{n+1} + 3x_n + x_{n-1})$$

Построить график передаточной функции и график отношения вычисляемого в результате фильтрации значения к истинному. Проинтерпретировать частотные свойства передаточной функции.

9. Провести сопоставительный анализ частотных характеристик передаточных функций, полученных при выполнении пп. 7 и 8.

Выполнение работы

1. Обозначим входной сигнал $s(t)$. После сглаживания полиномом первой степени, выходной сигнал будет иметь вид $y(t) = A + Bt$. Вычислим симметричный фильтр для m точек. Используя метод наименьших квадратов задача сводится к минимизации выражения:

$$f(A, B) = \sum_{k=-m}^{m}(s_k - y_k)^2 = \sum_{k=-m}^{m}(s_k - A - Bk)^2 \mapsto min$$

Для минимизации выражения примем частные производные равными 0. Вычислим частную производную по A:

$$\frac{\partial f(A, B)}{\partial A} = -2\sum_{k=-m}^{m}(s_k - A - Bk) = 0 \Rightarrow$$

$$\Rightarrow -\sum_{k=-m}^{m} s_k + \sum_{k=-m}^{m} A + \sum_{k=-m}^{m} Bk = 0$$

Отсюда можно выразить A, так как $\sum_{k=-m}^{m} Bk = 0$.

$$A = \frac{1}{2m+1} \sum_{k=-m}^{m} s_k$$

Тогда выходной сигнал в момент времени 0 имеет вид:

$$y(0) = A = \frac{1}{2m+1} \sum_{k=-m}^{m} s_k$$

В общем случае:

$$y(n) = \frac{1}{2m+1} \sum_{k=-m+n}^{m+n} s_k$$

Вычислим формулу передаточной функции фильтра, положив s_k равным $e^{j\omega k}$.

$$y(n) = \frac{1}{2m+1} \sum_{k=-m}^{m} e^{j\omega k} = H(\omega)$$

Используя следствие из формулы Эйлера $cosx = \frac{e^{jx}+e^{-jx}}{2}$ получаем

$$H(\omega) = \frac{1}{2m+1}(1 + \sum_{k=1}^{m} 2cos(k\omega))$$

Перейдем к $\tilde{H}(f) = H(2\pi f) = H(\omega)$ путем замены аргумента. Получаем следующие формулы для 3, 5, 7, 9 точек:

$$\tilde{H}_3(f) = \frac{1}{3}(1 + 2cos2\pi f)$$
$$\tilde{H}_5(f) = \frac{1}{5}(1 + 2cos2\pi f + 2cos4\pi f)$$
$$\tilde{H}_7(f) = \frac{1}{7}(1 + 2cos2\pi f + 2cos4\pi f + 2cos6\pi f)$$
$$\tilde{H}_9(f) = \frac{1}{9}(1 + 2cos2\pi f + 2cos4\pi f + 2cos6\pi f + 2cos8\pi f)$$

Отобразим полученные функции на интервале $f \in [0; 1]$ (см. рис. 1).

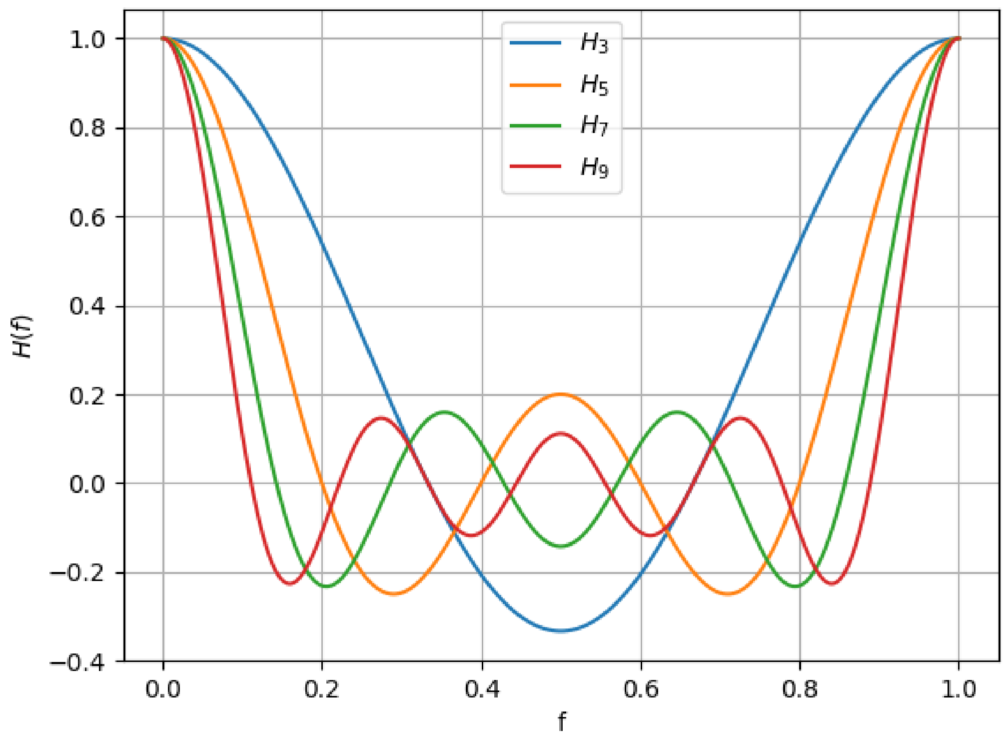

Рисунок 1 – Графики передаточных функций, полином первой степени

Можно заметить, что график передаточной функции, построенной по m точкам будет иметь m экстремумов.

2. После сглаживания полиномом второй степени, выходной сигнал будет иметь вид $y(t) = A + Bt + Ct^2$. Аналогично п. 1 минимизируем значение выражения

$$f(A, B) = \sum_{k=-m}^{m} (s_k - y_k)^2 = \sum_{k=-m}^{m} (s_k - A - Bk - Ck^2)^2 \mapsto min$$

Приравняем к нулю частные производные по A и C, получим

$$\begin{cases} \frac{\partial f(A,B,C)}{\partial A} = 0 \\ \frac{\partial f(A,B,C)}{\partial C} = 0 \end{cases} \Rightarrow \begin{cases} -2\sum_{k=-m}^{m}(s_k - A - Bk - Ck^2) = 0 \\ -2\sum_{k=-m}^{m} k^2(s_k - A - Bk - Ck^2) = 0 \end{cases} \Rightarrow$$

$$\Rightarrow \begin{cases} -\sum_{k=-m}^{m} s_k + \sum_{k=-m}^{m} A + \sum_{k=-m}^{m} Bk + \sum_{k=-m}^{m} Ck^2 = 0 \\ -\sum_{k=-m}^{m} k^2 s_k + \sum_{k=-m}^{m} Ak^2 + \sum_{k=-m}^{m} Bk^3 + \sum_{k=-m}^{m} Ck^4 = 0 \end{cases}$$

что приводит к системе уравнений, относительно A и C

$$\begin{cases} (2m+1)A + \frac{m(m+1)(2m+1)}{3}C = \sum_{k=-m}^{m} s_k \\ \frac{m(m+1)(2m+1)}{3}A + \frac{m(m+1)(2m+1)(3m^2+3m-1)}{15}C = \sum_{k=-m}^{m} k^2 s_k \end{cases}$$

откуда можно выразить A

$$A = \frac{\sum_{k=-m}^{m} k^2 s_k - \frac{3m^2+3m-1}{5}\sum_{k=-m}^{m} s_k}{\frac{m(m+1)(2m+1)}{3} - \frac{(3m^2+3m-1)(2m+1)}{5}}$$

Для сглаживания по 7 точкам формула будет иметь вид

$$y_7 = \frac{\sum_{k=-3}^{3} k^2 s_k - 7\sum_{k=-3}^{3} s_k}{28-49} = \frac{1}{21}(7\sum_{k=-3}^{3} s_k - \sum_{k=-3}^{3} k^2 s_k) =$$

$$= \frac{1}{21}(-2sn-3 + 3s_{n-2} + 6s_{n-1} + 7s_n + 6s_{n+1} + 3s_{n+2} - 2sn+3)$$

Вычислим формулу передаточной функции фильтра, положив s_k равным $e^{j\omega k}$. Группируя аналогично п.1 симметричные экспоненты в косинусы получаем:

$$\tilde{H}_7(f) = \frac{1}{21}(7 + 12cos2\pi f + 6cos4\pi f - 4cos6\pi f)$$

Выполним аналогичные вычисления для 9, 11 и 13. Формулы передаточных функций фильтра приведены ниже.

$$\tilde{H}_9(f) = \frac{1}{231}(59 + 108cos2\pi f + 78cos4\pi f + 28cos6\pi f - 42cos8\pi f)$$

$$\tilde{H}_{11}(f) = \frac{1}{429}(89 + 168cos2\pi f + 138cos4\pi f + 88cos6\pi f +$$
$$+18cos8\pi f - 72cos10\pi f)$$

$$\tilde{H}_{13}(f) = \frac{1}{143}(25 + 48cos2\pi f + 42cos4\pi f + 32cos6\pi f +$$
$$+18cos8\pi f - 22cos10\pi f)$$

Отобразим полученные функции на интервале $f \in [0; 1]$ (см. рис. 2).

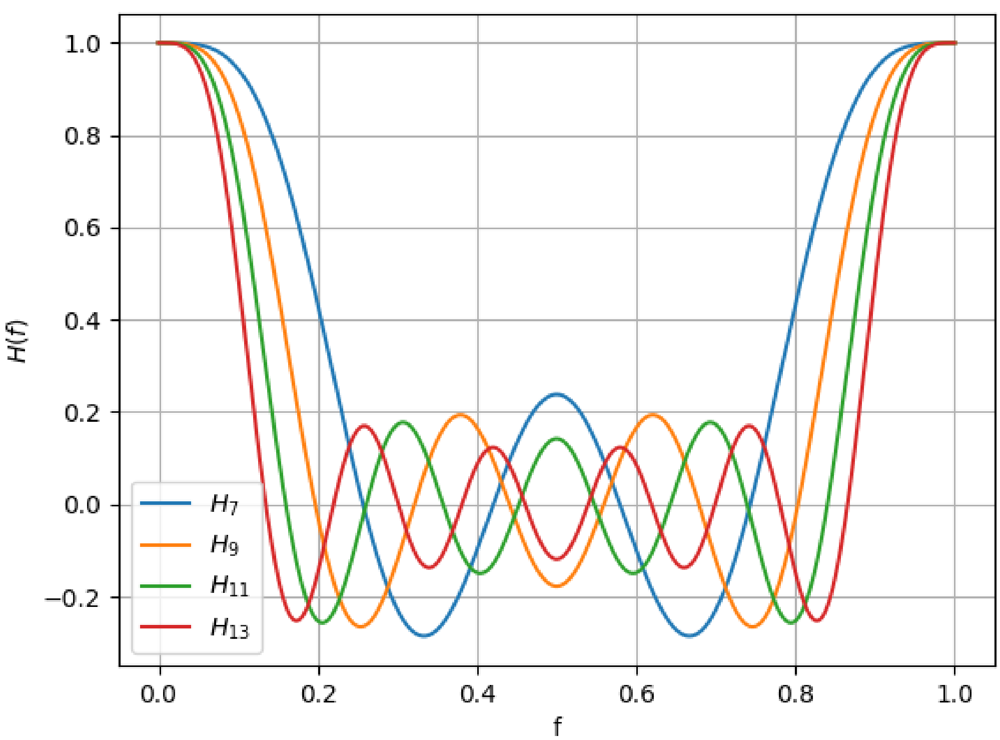

Рисунок 2 – Графики передаточных функций, полином второй степени

3. После сглаживания полиномом второй степени, выходной сигнал будет иметь вид $y(t) = A + Bt + Ct^2 + Dt^3 + Et^4$. С помощью МНК получим выражение для A, аналогично пунктам выше.

Проводя аналогичные п. 1 и 2 преобразования, приходим к следующей формуле вычисления выходного сигнала:

$$y(0) = 15(\frac{(12+5m(1+m)(-10+3m(1+m)))\sum_{k=-m}^{m} s_k}{4(-3+2m)(-1+2m)(1+2m)(3+2m)(5+2m)} - \\ -\frac{35(-3+2m(1+m))\sum_{k=-m}^{m} k^2 s_k - 63\sum_{k=-m}^{m} k^4 s_k}{4(-3+2m)(-1+2m)(1+2m)(3+2m)(5+2m)})$$

Для 9 точек формула будет иметь вид:

$$y_9(n) = \frac{1}{429}(15s_{n-4} - 55s_{n-3} + 30s_{n-2} + 135s_{n-1} + 179s_n + 135s_{n+1} + \\ +30s_{n+2} - 55s_{n+3} + 15s_{n+4})$$

После замены $s_k = e^{j\omega k}$ приходим к формуле передаточной функции для сглаживания по 9 точкам:

$$\tilde{H}_9(f) = \frac{1}{429}(179 + 270cos2\pi f + 60cos4\pi f - 110cos6\pi f + 30cos8\pi f)$$

Посчитаем аналогично для 11, 13 и 15 точек.

$$\tilde{H}_{11}(f) = \frac{1}{429}(143 + 240cos2\pi f + 120cos4\pi f - 20cos6\pi f - \\ -90cos8\pi f + 36cos10\pi f)$$

$$\tilde{H}_{13} = \frac{1}{2431}(677 + 1200cos2\pi f + 780cos4\pi f + 220cos6\pi f - \\ -270cos8\pi f - 396cos10\pi f + 220cos12\pi f)$$

$$\tilde{H}_{15}(f) = \frac{1}{46189}(11063 + 20250cos2\pi + 15000cos4\pi f + \\ +7510cos6\pi f - 330cos8\pi f - 5874cos10\pi f + \\ +5720cos12\pi f + 4290cos14\pi f)$$

Графики передаточных функций представлены на рисунке 3.

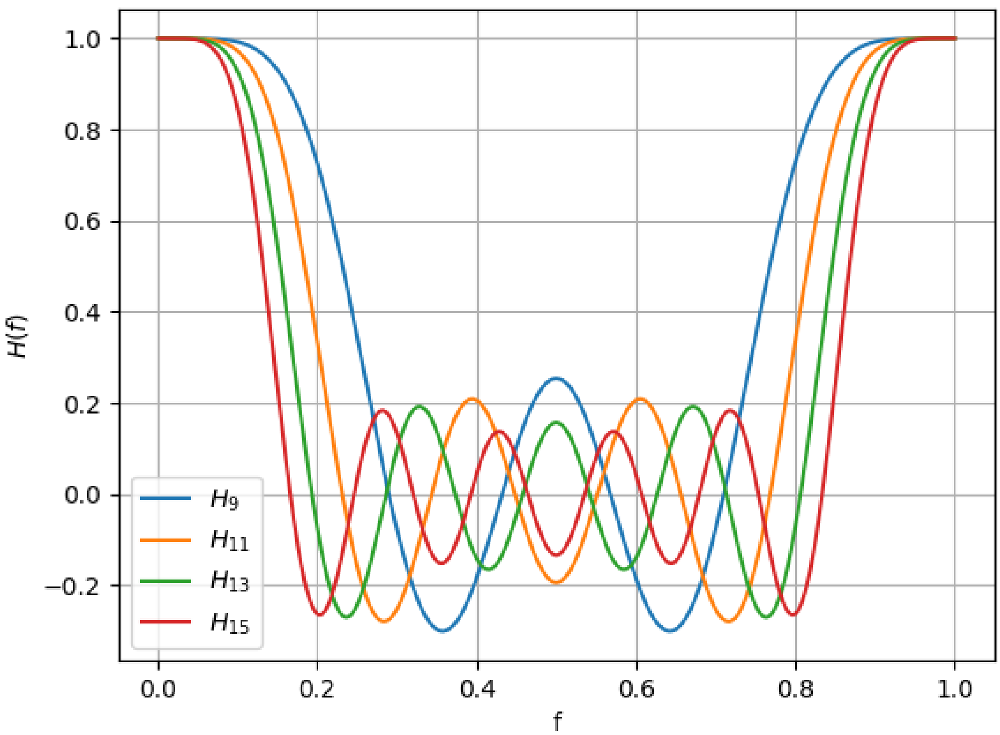

Рисунок 3 – Графики передаточных функций, полином четвертой степени

4. Из лекционных материалов были взяты формулы $y(n)$ для сглаживания по формуле Спенсера для 15 и 21 точек.

$$y_{15}(n) = \frac{1}{320}(-3s_{n-7} - 6s_{n-6} - 5s_{n-5} + 3s_{n-4} + 21s_{n-3} + 46s_{n-2} - 67s_{n-1} +$$
$$+74s_n + 67s_{n+1} + 46s_{n+2} + 21s_{n+3} + 3s_{n+4} - 5s_{n+5} - 6s_{n+6} - 3s_{n+7})$$

$$y_{21}(n) = \frac{1}{350}(-s_{n-10} - 3s_{n-9} - 5s_{n-8} - 5s_{n-7} - 2s_{n-6} + 6s_{n-5} + 18s_{n-4} +$$
$$+33s_{n-3} + 47s_{n-2} + 57s_{n-1} + 60s_n + 57s_{n+1} + 47s_{n+2} + 33s_{n+3} +$$
$$+18s_{n+4} + 6s_{n+5} - 2s_{n+6} - 5s_{n+7} - 5s_{n+8} - 3s_{n+9} - s_{n+10})$$

Соответствующие передаточные функции:

$$\tilde{H}_{15}(f) = \frac{1}{320}(74 + 134cos2\pi f + 92cos4\pi f + 42cos6\pi f + 6cos8\pi f -$$
$$-10cos10\pi f - 12cos12\pi f - 6cos14\pi f)$$

$$\tilde{H}_{21}(f) = \frac{1}{350}(60 + 114cos2\pi f + 94cos4\pi f + 66cos6\pi f +$$
$$36cos8\pi f + 12cos10\pi f - 4cos12\pi f - 10cos14\pi f - 10cos16\pi f -$$
$$-6cos18\pi f - 2cos20\pi f)$$

Графики передаточных функций представлены на рисунке 4.

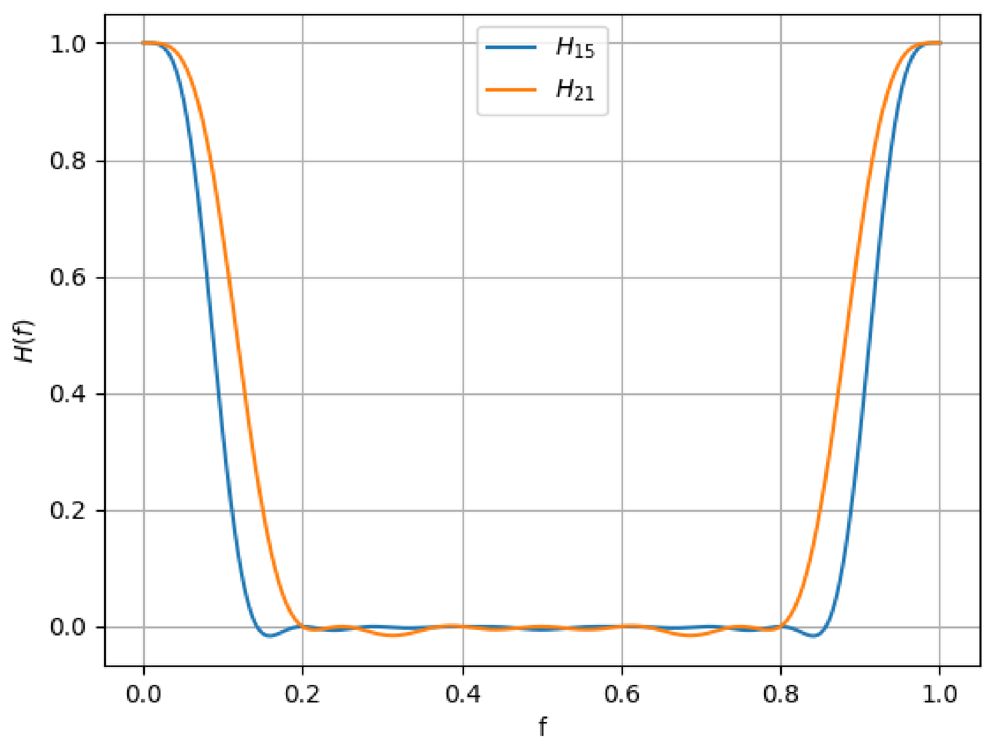

Рисунок 4 – Графики передаточных функций, формула Спенсера

5. «Шум» классически идентифицируется с высокими частотами, а «сообщение» или «информация» — с низкими частотами. Построим графики

рассмотренных передаточных функций в логарифмической шкале. Это позволит перемасштабировать график и более наглядно увидеть поведение передаточной функции на высоких и низких частотах, а также полосу пропускания. Эту величину называют логарифмической амплитудно-частотной характеристикой (ЛАХ):

$$H* = 20lgH$$

Графики рассмотренных передаточных функций, построенных в логарифмической шкале, приведены ниже.

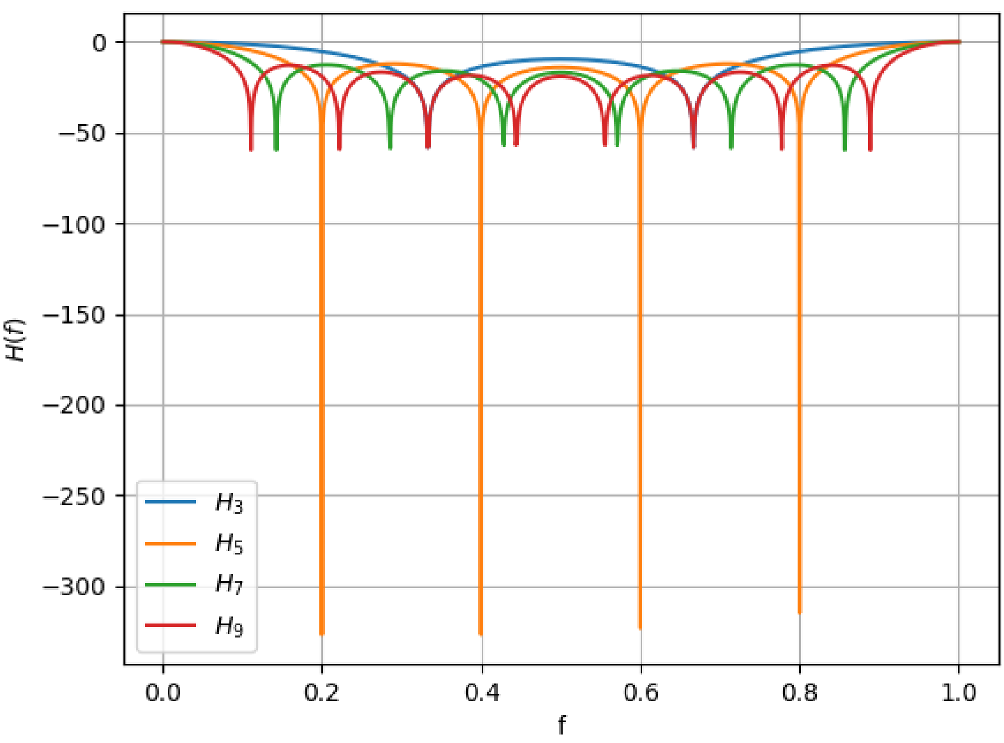

Рисунок 5 – Графики передаточных функций, полином первой степени, логарифмическая шкала

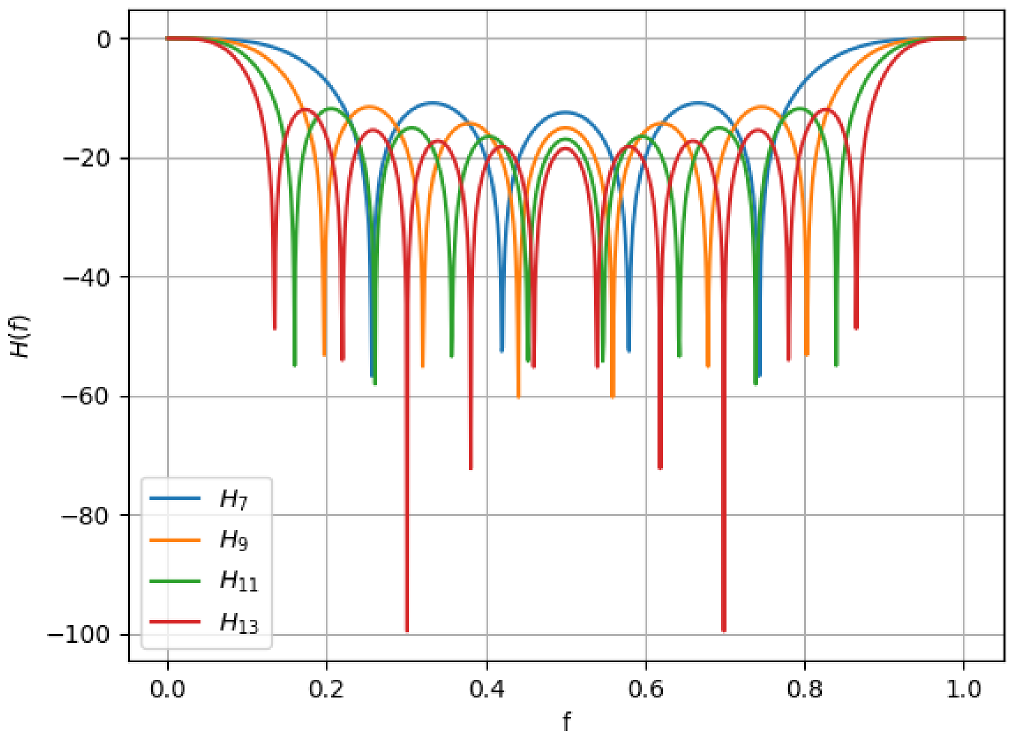

Рисунок 6 – Графики передаточных функций, полином второй степени, логарифмическая шкала

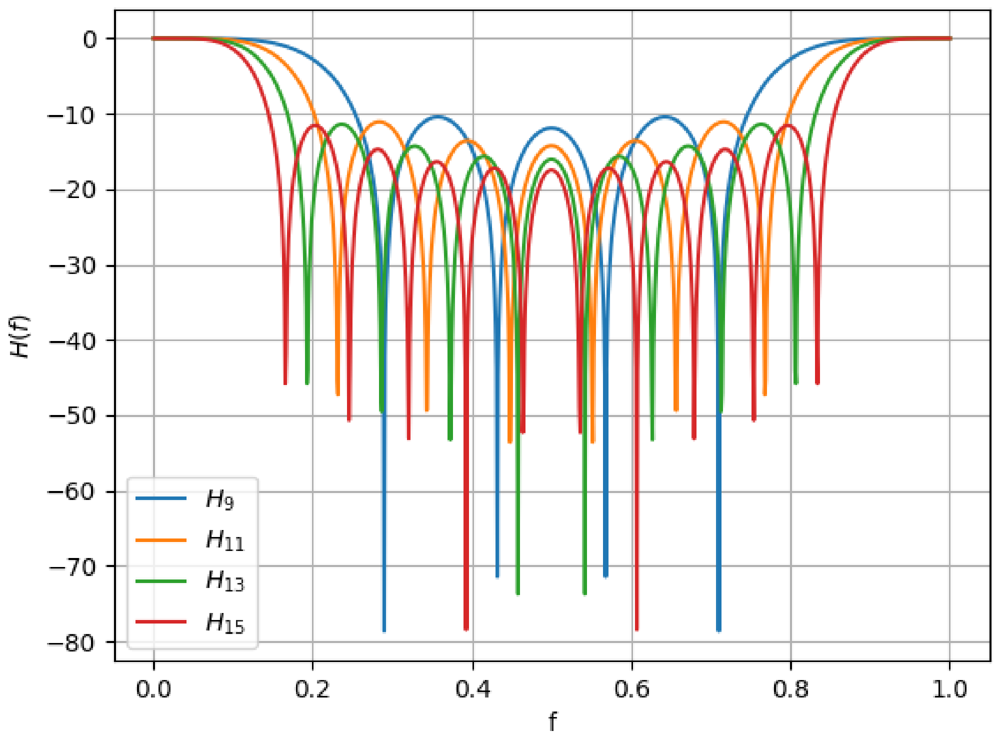

Рисунок 7 – Графики передаточных функций, полином четвертой степени, логарифмическая шкала

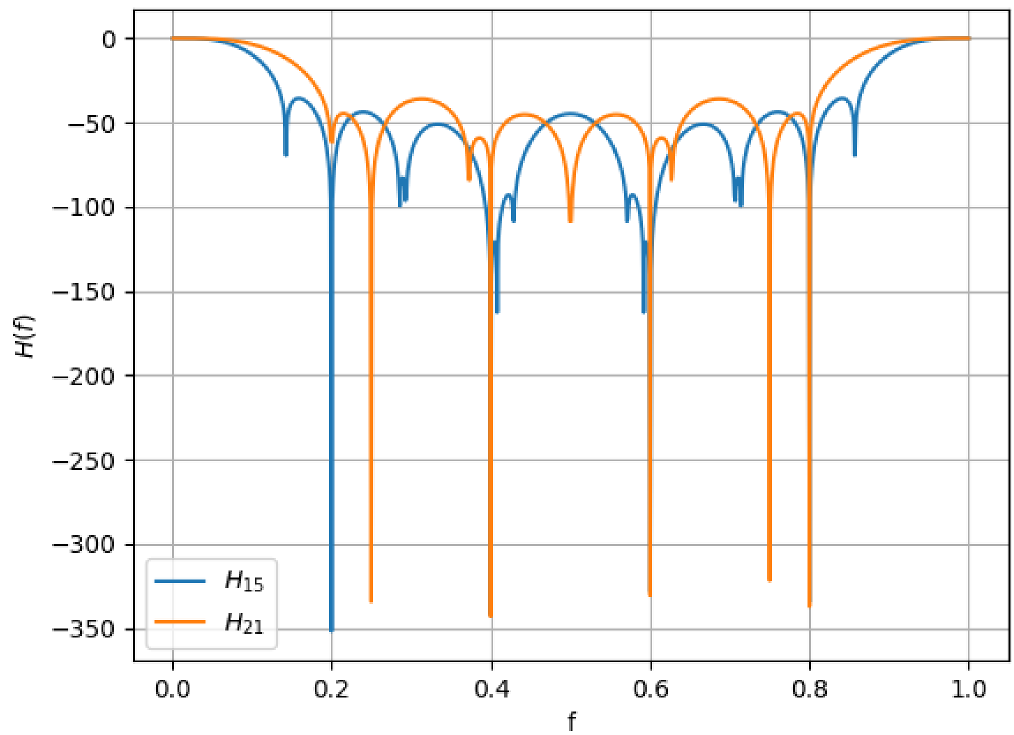

Рисунок 8 – Графики передаточных функций, формула Спенсера, логарифмическая шкала

6. Рассмотренные фильтры отличаются частотами среза и размерами полос пропускания. Для полиномиальных фильтров 1, 2 и 4 степени на графиках можно пронаблюдать, что чем больше точек используется для сглаживания, тем меньше размер полосы пропускания. От количества точек также зависят значения частот среза, поэтому выбирать фильтр и количество точек для сглаживания нужно в зависимости от поставленной задачи.

7. Выведем формулы передаточных функций рекурсивных фильтров, соответвующих формуле прямоугольников, формуле трапеций и формуле Симпсона.

Для формулы прямоугольников имеем следующее преобразование входного сигнала:

$$y_{n+1} = y_n + s_{n+\frac{1}{2}}$$

Пусть $s_n = e^{j\omega n}$ и $y_n = H(\omega)e^{j\omega n}$, тогда:

$$\begin{cases} y_{n+1} = H(\omega)e^{j\omega n} + e^{j\omega(n+\frac{1}{2})} \\ y_{n+1} = H(\omega)e^{j\omega(n+1)} \end{cases}$$

$$H(\omega)e^{j\omega n}e^{j\omega} = H(\omega)e^{j\omega n} + e^{j\omega n}e^{\frac{1}{2}j\omega}$$

$$H(\omega)(e^{j\omega n}e^{j\omega} - e^{j\omega n}) = e^{j\omega n}e^{\frac{1}{2}j\omega}$$

$$H(\omega)(e^{j\omega} - 1) = e^{\frac{1}{2}j\omega}$$

$$H(\omega) = \frac{1}{e^{\frac{j\omega}{2}} - e^{\frac{-j\omega}{2}}} = \frac{1}{2jsin\frac{\omega}{2}}$$

$$\tilde{H}(f) = \frac{1}{2jsin(\pi f)}$$

Для формулы трапеций:

$$y_{n+1} = y_n + \frac{1}{2}(s_n + s_{n+1})$$

$$\begin{cases} y_{n+1} = H(\omega)e^{j\omega n} + \frac{e^{j\omega n} + e^{j\omega(n+1)}}{2} \\ y_{n+1} = H(\omega)e^{j\omega(n+1)} \end{cases}$$

$$H(\omega)e^{j\omega n}e^{j\omega} = H(\omega)e^{j\omega n} + e^{j\omega n}\frac{1+e^{j\omega}}{2}$$

$$H(\omega)(e^{j\omega} - 1) = \frac{1+e^{j\omega}}{2}$$

$$H(\omega) = \frac{cos\frac{\omega}{2}}{2jsin(\frac{\omega}{2})}$$

$$\tilde{H}(f) = \frac{cos\pi f}{2jsin\pi f}$$

Для формулы Симпсона:

$$y_{n+1} = y_{n-1} + \frac{1}{3}(s_{n-1} + 4s_n + s_{n+1})$$

$$\begin{cases} y_{n+1} = H(\omega)e^{j\omega(n-1)} + \frac{e^{j\omega(n-1)}+4e^{j\omega n}+e^{j\omega(n+1)}}{3} \\ y_{n+1} = H(\omega)e^{j\omega(n+1)} \end{cases}$$

$$H(\omega)(e^{j\omega n}e^{j\omega}) = H(\omega)e^{j\omega n}e^{-j\omega} + e^{j\omega n}\frac{e^{-j\omega} + 4 + e^{j\omega}}{3}$$

$$H(\omega)(e^{j\omega} - e^{-j\omega}) = \frac{e^{-j\omega} + 4 + e^{j\omega}}{3} = \frac{cos\omega + 2}{3jsin\omega}$$

$$\tilde{H}(f) = \frac{cos2\pi f + 2}{3jsin2\pi f}$$

Графики передаточных функций приведены на рисунке 9.

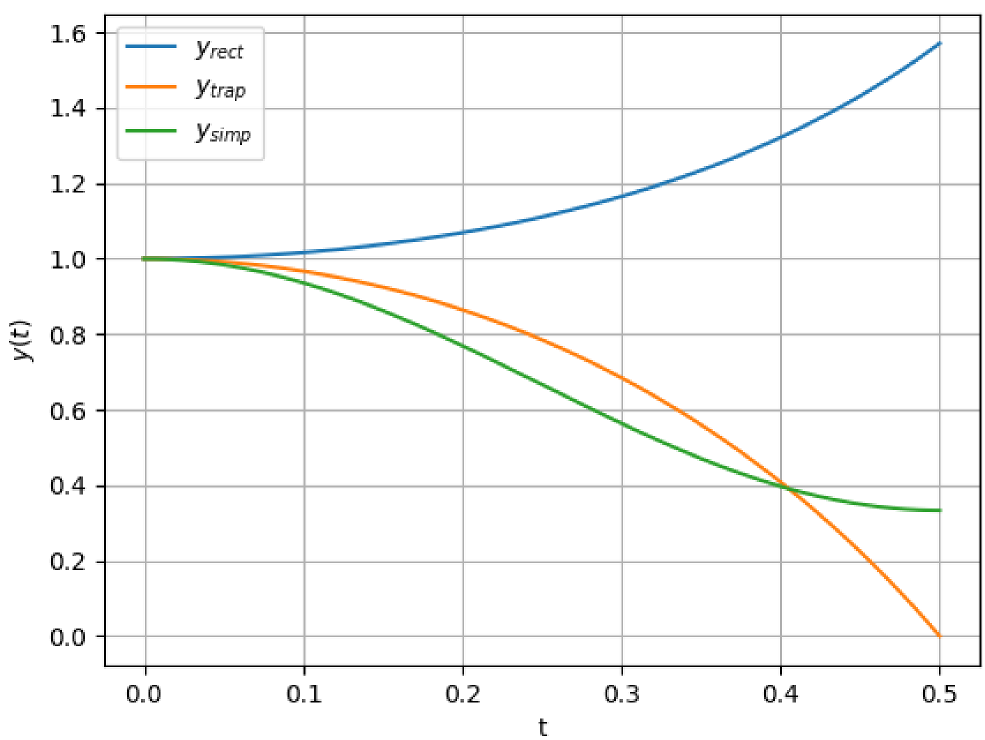

Рисунок 9 – Графики передаточных функций, формула прямоугольников, трапеций, Симпсона

Графики отношения вычисляемого в результате фильтрации значения к истинному для рассмотренных формул приведены на рисунках 10, 11 и 12.

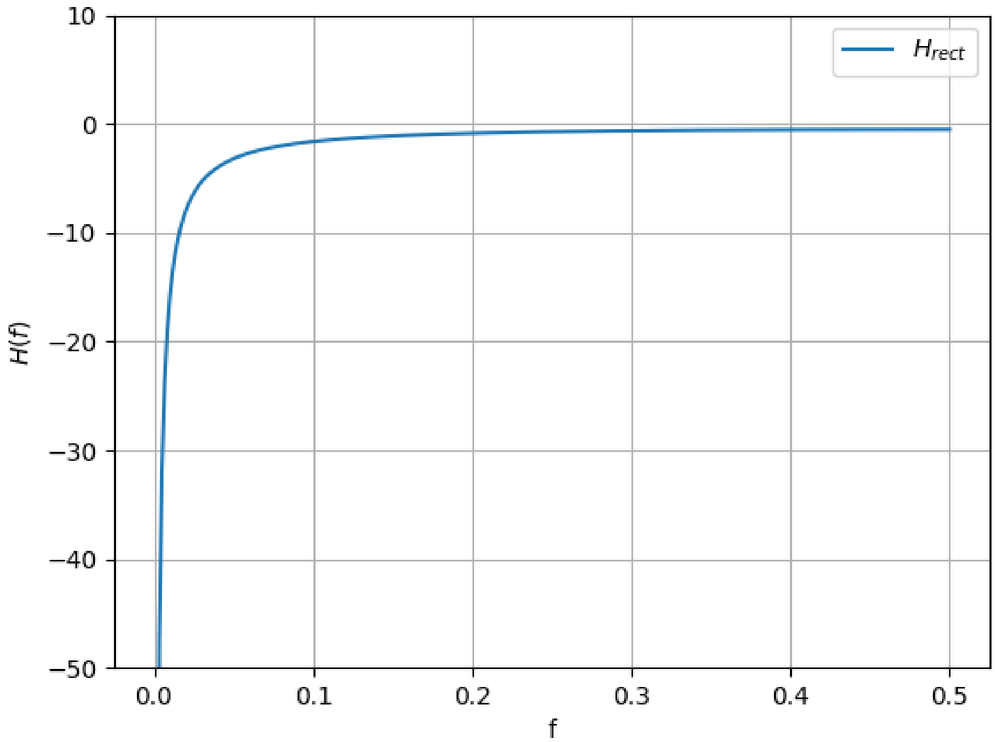

Рисунок 10 – График отношения вычисляемого значения к истинному, формула прямоугольников

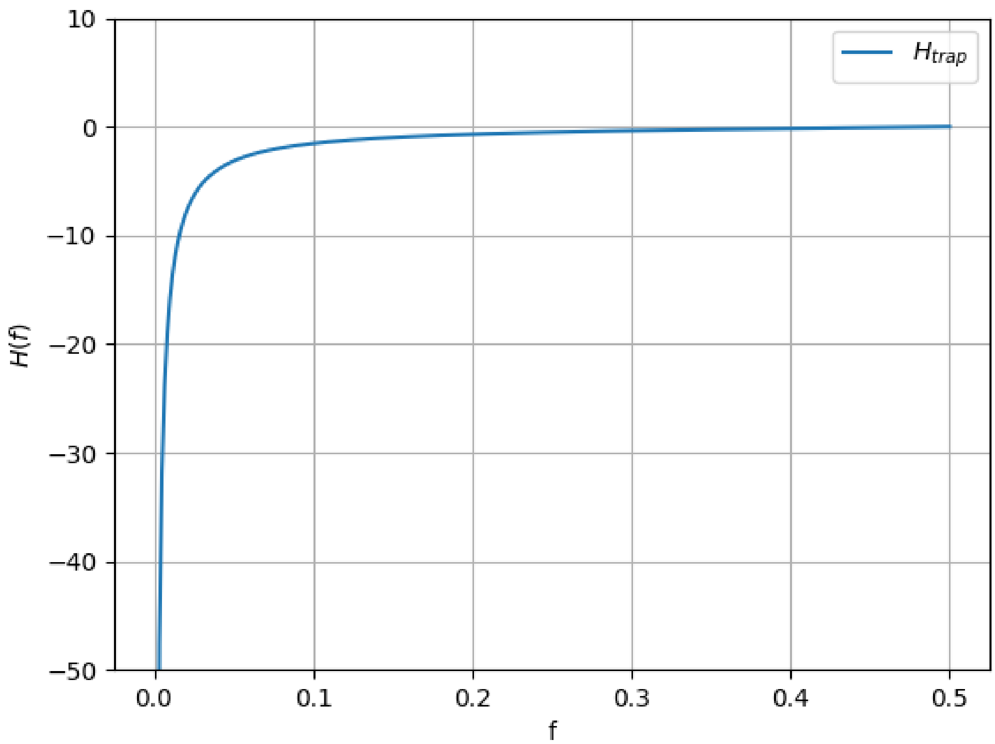

Рисунок 11 – График отношения вычисляемого значения к истинному, формула трапеций

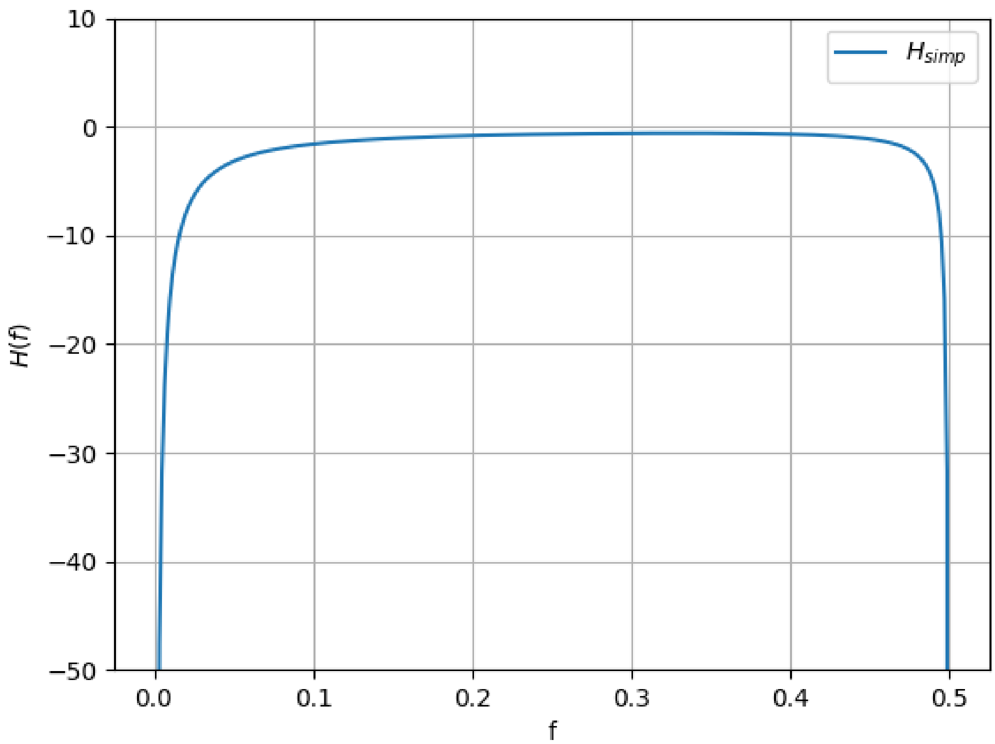

Рисунок 12 – График отношения вычисляемого значения к истинному, формула Симпсона

8. Выведем формулу передаточной функции рекурсивного фильтра, соответствующего квадратурной формуле.

Квадратурная формула преобразует сигнал следующим образом:

$$y_{n+2} = y_{n-1} + \frac{1}{8}(s_{n+2} + 3s_{n+1} + 3s_n + sn - 1)$$

$$\begin{cases} y_{n+2} = H(\omega)e^{j\omega(n-1)} + \frac{e^{j\omega(n+2)}+3e^{j\omega(n+1)}+3e^{j\omega n}+e^{j\omega(n-1)}}{8} \\ y_{n+2} = H(\omega)e^{j\omega(n+2)} \end{cases}$$

$$H(\omega)(e^{j\omega n}e^{2j\omega}) = H(\omega)e^{j\omega n}e^{-j\omega} + e^{j\omega n}\frac{e^{2j\omega} + 3e^{j\omega} + 3 + e^{-j\omega}}{8}$$

$$H(\omega)(e^{2j\omega} - e^{-j\omega}) = \frac{e^{2j\omega} + 3e^{j\omega} + 3 + e^{-j\omega}}{8}$$

$$H(\omega) = \frac{2cos\frac{3\omega}{2} + 6cos\frac{\omega}{2}}{16jsin\frac{3\omega}{2}}$$

$$\tilde{H}(f) = \frac{cos3\pi f + 3cos\pi f}{8jsin3\pi f}$$

График передаточной функции приведен на рисунке 13.

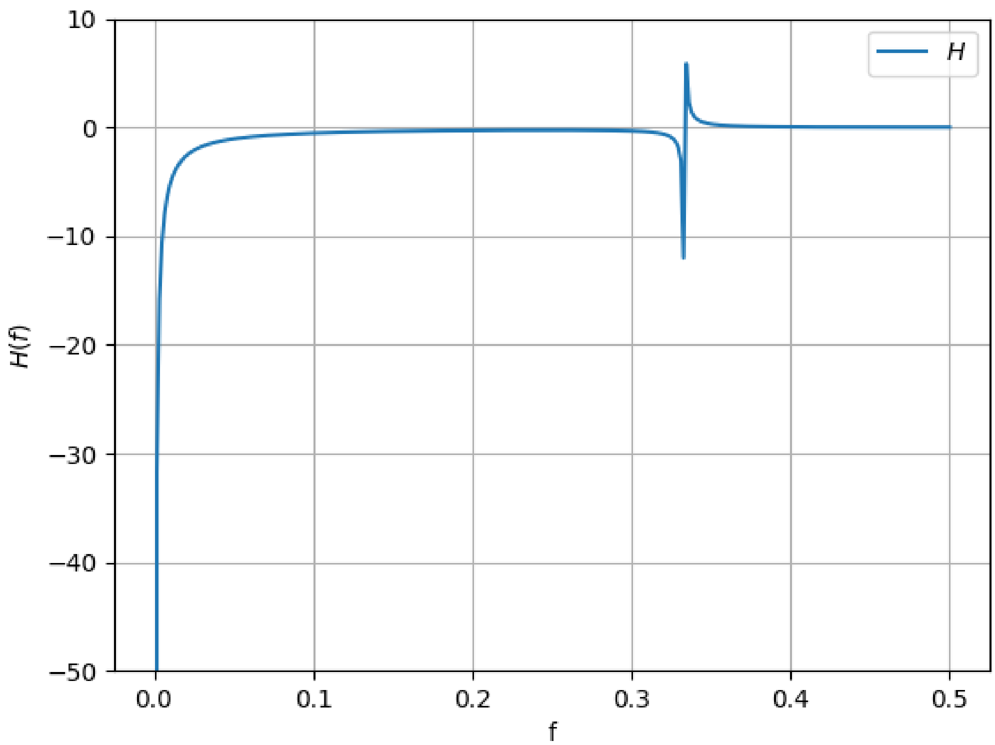

Рисунок 13 – График передаточной функции, квадратурная формула

Графики отношения вычисляемого в результате фильтрации значения к истинному представлен на рисунке 14.

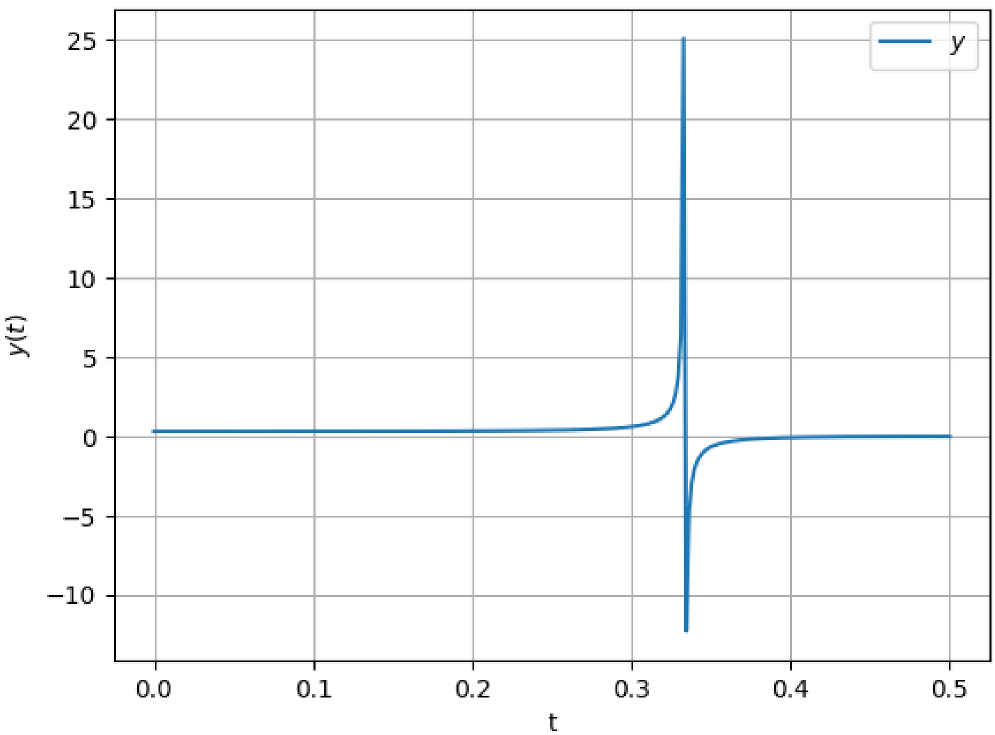

Рисунок 14 – График отношения вычисляемого значения к истинному, квадратурная формула

9. Формула прямоугольников завышает результаты на высоких частотах, а формула трапеций – занижает. Эти особенности легко объяснимы. Для одиночной гармоники площадь трапеции по двум последовательным отсчетам всегда меньше, чем площадь с выпуклой дугой гармоники между этими отсчетами, и разница тем больше, чем больше частота. Формула Симпсона отличается от формул трапеций и прямоугольников более высокой степенью касания единичного значения, что обеспечивает более высокую точность интегрирования в первой половине главного диапазона. Однако на высоких частотах погрешность начинает резко нарастать вплоть до выхода на бесконечность на конце диапазона.

У фильтра, соответствующего квадратурной формуле, очень широкая полоса пропускания и на определенной частоте наблюдается разрыв.

Выводы.

В результате выполнения работы были выведены формулы передаточных функций для фильтров, соответствующих полиномиальному сглаживанию 1, 2 и 4 степени с использованием методов наименьших квадратов. Также были выведены формулы передаточных функций для фильтров численного интегрирования.

По полученным формулам были построены графики и изучены свойства соответствующих фильтров.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ.

```
import math


import matplotlib.pyplot as plt
import numpy as np

f = np.linspace(0.001, 1.0, 1000)



def task1(fn=lambda x: x):
    h = lambda m: lambda x: np.array(
        [(1 + np.sum([2 * np.cos(2 * np.pi * m_ * x_) for m_ in
           range(1, m + 1)])) for x_ in x]) / (2 * m + 1)

    for i in range(1, 4 + 1):
        plt.plot(f, fn(h(i)(f)), label=rf'$H_{2 * i + 1}$')
    plt.xlabel('f')
    plt.ylabel(r'$H(f)$')
    plt.grid(True)
    plt.legend(loc="best")
    plt.show()



# task1()



def task2(fn=lambda x: x):
    h_7 = lambda x: 1 / 21 * (
            7 + 12 * np.cos(2 * np.pi * x) + 6 * np.cos(2 * np.pi
                * 2 * x) - 4 * np.cos(2 * np.pi * 3 * x))
    h_9 = lambda x: 1 / 231 * (59 + 108 * np.cos(2 * np.pi * x) +
        78 * np.cos(2 * np.pi * 2 * x) + 28 * np.cos(
```

```
        2 * np.pi * 3 * x) - 42 * np.cos(2 * np.pi * 4 * x))
    h_11 = lambda x: 1 / 429 * (
            89 + 168 * np.cos(2 * np.pi * x) + 138 * np.cos(2 *
               np.pi * 2 * x) + 88 * np.cos(
        2 * np.pi * 3 * x) + 18 * np.cos(2 * np.pi * 4 * x) - 72
           * np.cos(
        2 * np.pi * 5 * x))
    h_13 = lambda x: 1 / 143 * (
            25 + 48 * np.cos(2 * np.pi * x) + 42 * np.cos(2 * np.
               pi * 2 * x) + 32 * np.cos(
        2 * np.pi * 3 * x) + 18 * np.cos(2 * np.pi * 4 * x) - 22
           * np.cos(2 * np.pi * 6 * x))

    plt.plot(f, fn(h_7(f)), label=r'$H_7$')
    plt.plot(f, fn(h_9(f)), label=r'$H_9$')
    plt.plot(f, fn(h_11(f)), label=r'$H_{11}$')
    plt.plot(f, fn(h_13(f)), label=r'$H_{13}$')
    plt.xlabel('f')
    plt.ylabel(r'$H(f)$')
    plt.grid(True)
    plt.legend(loc="best")
    plt.show()


# task2()

def task3(fn=lambda x: x):
    h_9 = lambda x: 1 / 429 * (179 + 270 * np.cos(2 * math.pi * x
       ) + 60 * np.cos(2 * math.pi * 2 * x) - 110 * np.cos(
        2 * math.pi * 3 * x) + 30 * np.cos(2 * math.pi * 4 * x))
    h_11 = lambda x: 1 / 429 * (
            143 + 240 * np.cos(2 * math.pi * x) + 120 * np.cos(2
```

```
                * math.pi * 2 * x) - 20 * np.cos(
            2 * math.pi * 3 * x) - 90 * np.cos(2 * math.pi * 4 * x) +
                36 * np.cos(
            2 * math.pi * 5 * x))
    h_13 = lambda x: 1 / 2431 * (
                677 + 1200 * np.cos(2 * math.pi * x) + 780 * np.cos(2
                    * math.pi * 2 * x) + 220 * np.cos(
            2 * math.pi * 3 * x) - 270 * np.cos(2 * math.pi * 4 * x)
               - 396 * np.cos(
            2 * math.pi * 5 * x) + 220 * np.cos(2 * math.pi * 6 * x))
    h_15 = lambda x: 1 / 46189 * (
                11063 + 20250 * np.cos(2 * math.pi * x) + 15000 * np.
                   cos(2 * math.pi * 2 * x) + 7510 * np.cos(
            2 * math.pi * 3 * x) - 330 * np.cos(
            2 * math.pi * 4 * x) - 5874 * np.cos(2 * math.pi * 5 * x)
                - 5720 * np.cos(2 * math.pi * 6 * x) + 4290 * np.cos(
            2 * math.pi * 7 * x))

    plt.plot(f, fn(h_9(f)), label=r'$H_9$')
    plt.plot(f, fn(h_11(f)), label=r'$H_{11}$')
    plt.plot(f, fn(h_13(f)), label=r'$H_{13}$')
    plt.plot(f, fn(h_15(f)), label=r'$H_{15}$')
    plt.xlabel('f')
    plt.ylabel(r'$H(f)$')
    plt.grid(True)
    plt.legend(loc="best")
    plt.show()


# task3()
```

```python
def task4(fn=lambda x: x):
    h_21 = lambda x: 1 / 320 * (74 + 134 * np.cos(2 * math.pi * x
       ) + 92 * np.cos(4 * math.pi * x) + 42 * np.cos(
        6 * math.pi * x) + 6 * np.cos(8 * math.pi * x) - 10 * np.
           cos(10 * math.pi * x) - 12 * np.cos(
        12 * math.pi * x) - 6 * np.cos(14 * math.pi * x))
    h_15 = lambda x: 1 / 350 * (60 + 114 * np.cos(2 * math.pi * x
       ) + 94 * np.cos(4 * math.pi * x) + 66 * np.cos(
        6 * math.pi * x) + 36 * np.cos(8 * math.pi * x) + 12 * np
           .cos(10 * math.pi * x) - 4 * np.cos(
        12 * math.pi * x) - 10 * np.cos(14 * math.pi * x) - 10 *
           np.cos(16 * math.pi * x) - 6 * np.cos(
        18 * math.pi * x) - 2 * np.cos(20 * math.pi * x))

    plt.plot(f, fn(h_15(f)), label=r'$H_{15}$')
    plt.plot(f, fn(h_21(f)), label=r'$H_{21}$')
    plt.xlabel('f')
    plt.ylabel(r'$H(f)$')
    plt.grid(True)
    plt.legend(loc="best")
    plt.show()


# task4()

def task5():
    log_scaler = lambda x: 20 * np.log10(np.abs(x))
    task1(log_scaler)
    task2(log_scaler)
    task3(log_scaler)
    task4(log_scaler)
```

```
# task5()


def task7():
    h_rect = lambda x: (1 / (2j * np.sin(math.pi * x))).imag
    h_trap = lambda x: (np.cos(math.pi * x) / (2j * np.sin(math.
       pi * x))).imag
    h_simp = lambda x: ((np.cos(2 * math.pi * x) + 2) / (3j * np.
       sin(2 * math.pi * x))).imag

    y_rect = lambda x: math.pi * x / (np.sin(math.pi * x))
    y_trap = lambda x: np.cos(math.pi * x) * (math.pi * x / np.
       sin(x * math.pi))
    y_simp = lambda x: (np.cos(2 * math.pi * x) + 2) / 3

    t = np.linspace(1e-10, 0.5, 300)

    plt.plot(t, h_rect(t), label=r'$H_{rect}$')
    plt.xlabel('f')
    plt.ylabel(r'$H(f)$')
    plt.grid(True)
    plt.legend(loc="best")
    x1, x2, y1, y2 = plt.axis()
    plt.axis((x1, x2, -50, 10.))
    plt.show()

    plt.plot(t, h_trap(t), label=r'$H_{trap}$')
    plt.xlabel('f')
    plt.ylabel(r'$H(f)$')
    plt.grid(True)
    plt.legend(loc="best")
```

```
    x1, x2, y1, y2 = plt.axis()
    plt.axis((x1, x2, -50, 10.))
    plt.show()

    plt.plot(t, h_simp(t), label=r'$H_{simp}$')
    plt.xlabel('f')
    plt.ylabel(r'$H(f)$')
    plt.grid(True)
    plt.legend(loc="best")
    x1, x2, y1, y2 = plt.axis()
    plt.axis((x1, x2, -50, 10.))
    plt.show()

    plt.plot(t, y_rect(t), label=r'$y_{rect}$')
    plt.plot(t, y_trap(t), label=r'$y_{trap}$')
    plt.plot(t, y_simp(t), label=r'$y_{simp}$')
    plt.xlabel('t')
    plt.ylabel(r'$y(t)$')
    plt.grid(True)
    plt.legend(loc="best")
    plt.show()


# task7()



def task8():
    h = lambda x: ((np.cos(3 * math.pi * x) + 3 * np.cos(math.pi
       * x)) / (8j * np.sin(3 * math.pi * x))).imag
    y = lambda x: (1 / 12) * (np.cos(3 * math.pi * x) + 3 * np.
       cos(math.pi * x)) * (
            (3 * math.pi * x) / np.sin(3 * math.pi * x))
```

```
    t = np.linspace(1e-10, 0.5, 300)

    plt.plot(t, h(t), label=r'$H$', )
    plt.xlabel('f')
    plt.ylabel(r'$H(f)$')
    plt.grid(True)
    plt.legend(loc="best")
    x1, x2, y1, y2 = plt.axis()
    plt.axis((x1, x2, -50, 10.))
    plt.show()

    plt.plot(t, y(t), label=r'$y$')
    plt.xlabel('t')
    plt.ylabel(r'$y(t)$')
    plt.grid(True)
    plt.legend(loc="best")
    plt.show()


task8()
```