Определение требований к алгоритму цифровой обработки сигнала. Теоретические сведения

В работе будет использован БИХ фильтр. Причина выбора именно этого типа в том, что на фильтрацию входного сигнала уходит меньше процессорного времени, по сравнению с КИХ фильтром.

Общая идея всех методов расчёта рекурсивных фильтров состоит в том, чтобы, используя передаточную функцию аналогового фильтра-прототипа W(p), получить дискретную передаточную функцию ЦФ D(z). При этом заранее нужно выбрать значение периода дискретизации T_Δ, исходя из требований теоремы Котельникова, и, кроме того, желательно, чтобы верхняя граничная частота полосы пропускания фильтра была бы как минимум в несколько раз (лучше – на порядок) меньше частоты Найквиста.

Переход от W(p) к D(z) рекурсивного ЦФ можно осуществить с использованием любого из трех основных методов:

1) метод инвариантности переходной характеристики;

2) метод на основе формул дискретного интегрирования;

3) метод согласованного Z-преобразования.

Наибольшее распространение на практике получил метод на основе формул дискретного интегрирования, который использует связь между переменными p и z.[1] Именно таким мы и собираемся воспользоваться, а конкретнее – методом обобщённого билинейного преобразования. Он заключается в замене p в передаточной функции нормированного аналогового ФНЧ прототипа формулой с z оператором, причём для каждого типа фильтра формала будет отличаться. Формулы замены переменной в таблице 1

Таблица 1 − Формулы замены переменной р, применяемые при преобразовании передаточной функции нормированного аналогового ФНЧ-прототипа в передаточную функцию цифрового фильтра заданного типа методом обобщенного билинейного преобразования.

Тип	Параметры результирующего ЦФ	Формула для замены переменной*
ФНЧ	Частота среза $\omega_{ср}$	$p \to \frac{2}{\omega_{cp} T_\Delta} \frac{z-1}{z+1}$
ФВЧ	Частота среза $\omega_{ср}$	$p \to \frac{\omega_{cp} T_\Delta}{2} \frac{z+1}{z-1}$
ПФ	Граничные частоты полосы пропускания $\omega_н$ и $\omega_в$	$p \to \gamma \frac{z^2 - 2\zeta z + 1}{z^2 - 1}$
РФ	Граничные частоты полосы пропускания $\omega_н$ и $\omega_в$	$p \to \frac{z^2 - 1}{\gamma (z^2 - 2\zeta z + 1)}$

* Примечание $\gamma = ctg\left(\frac{T_\Delta}{2}(\omega_в - \omega_н)\right)$, $\zeta = \frac{\cos\left(\frac{T_\Delta}{2}(\omega_в + \omega_н)\right)}{\cos\left(\frac{T_\Delta}{2}(\omega_в - \omega_н)\right)}$.

Помимо прямой замены есть способ посчитать и АЧХ фильтра и его коэффициенты по всё тому же методу билинейного преобразования. В таблице 2 содержатся формулы, которые позволят после расчёта осуществить фильтрацию по фомруле 1

$$y_n = \sum_{k=0}^{q} b_k x_{n-k} - \sum_{m=1}^{r} a_m y_{n-m} \qquad (1)$$

Таблица 2 − Формулы сомножителей D к (z), получающиеся при применении формул обобщённого билинейного преобразования к выражению (2)

Тип ЦФ	Формула для $D_К(z)$	Формулы* для коэффициентов $D_К(z)$
ФНЧ с частотой среза $\omega_{ср}$	$\frac{\beta_0 z^2 + \beta_1 z + \beta_2}{\alpha_0 z^2 + \alpha_1 z + \alpha_2}$	$\beta_0 = (\omega_{cp} T_\Delta)^2, \beta_1 = 2\beta_0, \beta_2 = \beta_0,$ $\alpha_0 = a_2 (\omega_{cp} T_\Delta)^2 + 2 a_1 \omega_{cp} T_\Delta + 4 a_0,$ $\alpha_1 = 2 a_2 (\omega_{cp} T_\Delta)^2 - 8 a_0,$ $\alpha_2 = a_2 (\omega_{cp} T_\Delta)^2 - 2 a_1 \omega_{cp} T_\Delta + 4 a_0$

ФВЧ с частотой среза $\omega_{ср}$	$\frac{\beta_0 z^2+\beta_1 z+\beta_2}{\alpha_0 z^2+\alpha_1 z+\alpha_2}$	$\beta_0=4, \beta_1=-8, \beta_2=4,$ $\alpha_0=a_0(\omega_{cp} T_\Delta)^2+2a_1\omega_{cp} T_\Delta+4a_2,$ $\alpha_1=2a_0(\omega_{cp} T_\Delta)^2-8a_2,$ $\alpha_2=a_0(\omega_{cp} T_\Delta)^2-2a_1\omega_{cp} T_\Delta+4a_2$
ПФ с частотами среза $\omega_н$ и $\omega_в$	$\frac{\beta_0 z^4+\beta_1 z^3+\beta_2 z^2+\beta_3 z+\beta_4}{\alpha_0 z^4+\alpha_1 z^3+\alpha_2 z^2+\alpha_3 z+\alpha_4}$	$\beta_0=1, \beta_1=0, \beta_2=-2, \beta_3=0, \beta_4=1,$ $\alpha_0=\gamma^2 a_0+\gamma a_1+a_2,$ $\alpha_1=-4\gamma^2\zeta a_0-2\gamma\zeta a_1,$ $\alpha_2=4\gamma^2\zeta^2 a_0+2\gamma a_0-2a_2,$ $\alpha_3=-4\gamma^2\zeta a_0+2\gamma\zeta a_1,$ $\alpha_4=\gamma^2 a_2-\gamma a_1+a_0$
РФ с частотами среза $\omega_н$ и $\omega_в$	$\frac{\beta_0 z^4+\beta_1 z^3+\beta_2 z^2+\beta_3 z+\beta_4}{\alpha_0 z^4+\alpha_1 z^3+\alpha_2 z^2+\alpha_3 z+\alpha_4}$	$\beta_0=\gamma^2, \beta_1=-4\gamma^2\zeta, \beta_2=4\gamma^2\zeta^2+2\gamma^2,$ $\beta_3=\beta_1, \beta_4=\beta_0,$ $\alpha_0=\gamma^2 a_2-\gamma a_1+a_2,$ $\alpha_1=4\gamma^2\zeta a_2-2\gamma\zeta a_1,$ $\alpha_2=4\gamma^2\zeta^2 a_2+2\gamma a_2-2a_0,$ $\alpha_3=-4\gamma^2\zeta a_2+2\gamma\zeta a_1,$ $\alpha_4=\gamma^2 a_2-\gamma a_1+a_0$

Для того, чтобы посчитать коэффициенты выше понадобятся а-коэффициенты. Они получаются из коэффициентов передаточной функции нормированного ФНЧ прототипа. Можно выделить несколько видов фильтров:

- Бесселя
- Чебышева
- Баттерворта

У них есть характеристические отличия, но в данном случае нас интересует только то, что фильтр Баттерворта наиболее просто рассчитывается. Его передаточная функция нормированного ФНЧ-прототипа показана в формуле (2)

$$W_{Bt}(p)=\begin{cases} \dfrac{1}{\prod\limits_{k=1}^{\frac{n}{2}} p^2-2p\cos\left(\dfrac{2k+n-1}{2n}\pi\right)+1}, для\,чётных\,n \\ \dfrac{1}{\prod\limits_{k=1}^{\frac{n}{2}}\left(p^2-2p\cos\left(\dfrac{2k+n-1}{2n}\pi\right)+1\right)(p+1)}, для\,нечётных\,n \end{cases} \qquad (2)$$

где n – порядок фильтра

Выдвинем гипотезу, что 2-ой порядок данного типа фильтра соответствует ТЗ. Для проверки этой гипотезы смоделируем фильтр. Программа для моделирования написана на python 3.7 и её полная версия написана в приложении 1. Прокомментирую наиболее важные части – функцию recursive_transfer_function и generate_recursive_filter. Сначала рассмотрим первую. В ней непосредственно производится обобщённое билинейное преобразование.

На вход функция принимает значение z оператора и нижние и верхние частоты среза в герцах. Первое, что в ней делается – перевод частот в радианы:

```
fn = fn * 2 * np.pi
fv = fv * 2 * np.pi
```

Затем вводятся константы записанные в примечании таблицы 1

```
gamma = 1/np.tan((delta_T/2) * (fv - fn))
zeta = np.cos((delta_T/2) * (fv + fn))/np.cos((delta_T/2) * (fv - fn))
```

После чего выписывается формула замены переменной из таблицы 1 соответствующая ПФ:

```
changed_p = gamma * (((z**2) - 2 * zeta * z + 1)/((z**2) - 1))
```

Ну и наконец передаём это всё в передаточную функцию Баттерворта 2-ого порядка полученной из формулы (2):

```
calculated_transfer_function = 1/((changed_p**2) + 1.41421 * changed_p + 1)
```

Результат этой формулы возвращаем из функции.

Соответственно, чтобы получить коэффициент подавления на конкретной частоте нужно передавать на место z оператора результат формулы $e^{2\pi f T_{\Delta} j}$, где f – частота для которой проводится расчёт, и возвести получившееся комплексное число в модуль. Чтобы отобразить результат в дБ нужно взять от получившегося числа логарифм по основанию 10 и умножить на 20. В результате получается график АЧХ фильтра, показанный на рисунке 1

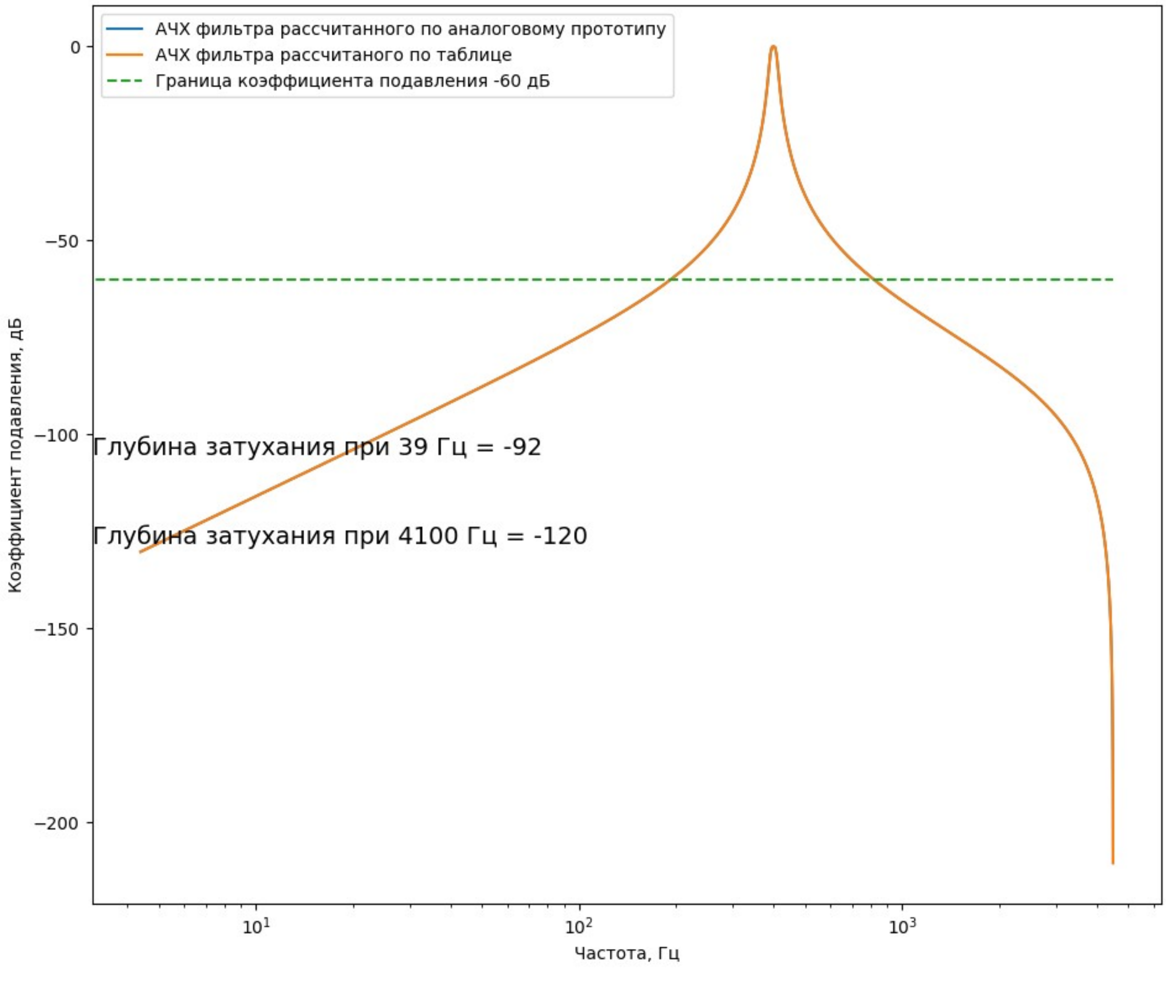

Рисунок 1 – АЧХ фильтра

Далее, рассмотрим функцию generate_recursive_filter. Эта функция считает коэффициенты уже по таблице, без непосредственной подстановки. На вход функция принимает значение z оператора и нижние и верхние частоты среза в герцах. Снова осуществляем перевод частот в радианы:

```
fn = fn * 2 * np.pi
fv = fv * 2 * np.pi
```

Затем вводятся константы записанные в примечании таблицы 1

```
gamma = 1/np.tan((delta_T/2) * (fv - fn))
zeta = np.cos((delta_T/2) * (fv + fn))/np.cos((delta_T/2) * (fv - fn))
```

Далее определяются те самые а-коэффициенты

```
a_constants = [1, -2 * np.cos(((2 * order_number + filter_order - 1)/(2 * filter_order)) * np.pi), 1]
```

После чего считаются коэффициенты α и β по таблице 2

```
current_betta = [1, 0, -2, 0, 1]

current_alpha = [gamma**2 * a_constants[0] + gamma * a_constants[1] + a_constants[2],
                 -4 * gamma**2 * zeta * a_constants[0] - 2 * gamma * zeta * a_constants[1],
                 4 * gamma**2 * zeta**2 * a_constants[0] + 2 * gamma**2 * a_constants[0] - 2 * a_constants[2],
                 -4 * gamma**2 * zeta * a_constants[0] + 2 * gamma * zeta * a_constants[1],
                 gamma**2 * a_constants[0] - gamma * a_constants[1] + a_constants[2]]
```

Теперь считаем $D_k(z)$ с полученными коэффициентами

```
current_betta = [1, 0, -2, 0, 1]

current_alpha = [gamma**2 * a_constants[0] + gamma * a_constants[1] + a_constants[2],
                 -4 * gamma**2 * zeta * a_constants[0] - 2 * gamma * zeta * a_constants[1],
                 4 * gamma**2 * zeta**2 * a_constants[0] + 2 * gamma**2 * a_constants[0] - 2 * a_constants[2],
                 -4 * gamma**2 * zeta * a_constants[0] + 2 * gamma * zeta * a_constants[1],
                 gamma**2 * a_constants[0] - gamma * a_constants[1] + a_constants[2]]
```

Далее высчитываем дискретную передаточную функцию. С помощью получившихся коэффициентов

```
our_filter = our_filter * ((current_betta[0] * z**4 + current_betta[1] * z**3 + current_betta[2] * z**2 + current_betta[3] * z + current_betta[4])/ \
    (current_alpha[0] * z**4 + current_alpha[1] * z**3 + current_alpha[2] * z**2 + current_alpha[3] * z + current_alpha[4]))
```

Наконец вычисляем коэффициенты α и β для фильтрации. Результат вычисления последних можно увидеть на рисунке 2

```
alphas = np.array(current_alpha)/current_alpha[0]
betas = np.array(current_betta)/current_alpha[0]
```

```
  calculated_transfer_function = 1/((changed_p**2) + 1.41421 * changed_p + 1)
Альфа коэффициенты
1-ый: 1.0
2-ый: -3.8261594180878564230852134642191231250762939453125
3-ый: 5.6402167766250270730665761220734566450119018554687 5
4-ый: -3.788568424693382841184075005003251135349273681640625
5-ый: 0.98044753188059352577710114928777329623699188232421875
Бета коэффициенты
1-ый: 4.82615212977725109556002835997645661336719058454036712646484375e-05
2-ый: 0.0
3-ый: -9.6523042595545021911200567199529132267343811690807342529296875e-05
4-ый: 0.0
5-ый: 4.82615212977725109556002835997645661336719058454036712646484375e-05
```

Рисунок 2 – Коэффициенты фильтра

И без вычислений коэффициента затухания на частотах 39 Гц и 4100 Гц рисунка 1 заметно, что они ниже заданных по заданию -60 дБ, более точные коэффициенты затухания написаны на рисунке. Следовательно, гипотеза подтверждена и фильтр 2-ого порядка нам подходит.

Из рассчитанных коэффициентов α и β, продемонстрированных на рис 2 можно составить разностное уравнение, округляя до количества чисел после запятой, чуть большей минимума (12 знаков), ради простоты записи

$$y_k = 4.82615212978 \cdot 10^{-5} x_k - 9.65230425955 \cdot 10^{-5} x_{k-2} + 4.82615212978 \cdot 10^{-5} x_{k-4} -$$
$$+ 3.82615941809\, y_{k-1} - 5.64021677663\, y_{k-2} + 3.78856842469\, y_{k-3} - -3.78856842469\, y_{k-4}$$

И без вычислений коэффициента затухания на частотах 39 Гц нои 4100 Гц заметно, что они ниже заданных по заданию -60 дБ, более точные коэффициенты написаны на рисунке. Следовательно, гипотеза подтверждена и фильтр 2-ого порядка нам подходит.

Список литературы:

1. Цифровые фильтры частотной селекции: учебное пособие / О.О. Жаринов, И.О. Жаринов.СПб: Изд-во ГУАП, 2019. – 77 с. [библиотечный шифр 621.372 Ж 34].

ПРИЛОЖЕНИЕ 1

```
# Const
delta_T = 1/9000
q = 2048


def generate_recursive_filter(z, fn=0, fv=1):
    """
        Генерирует рекурсивный фильтр с заданными параметрами и его коэффициенты

    :z: оператор передаточной функции
    :fn: Нижняя граница частоты среза фильтра
    :fv: Верхняя граница частоты среза фильтра
    :returns: Теоритический фильтр по аналоговой передаточной функции
    :returns: Alpha коэффициенты фильтра
    :returns: Beta коэффициенты фильтра
    """
    import numpy as np

    # Преобразовываем частоты для правильной работы
    fn = fn * 2 * np.pi
    fv = fv * 2 * np.pi

    gamma = 1/np.tan((delta_T/2) * (fv - fn))
    zeta = np.cos((delta_T/2) * (fv + fn))/np.cos((delta_T/2) * (fv - fn))

    # переменные используемые по формуле
    filter_order = 2
    order_number = 1

    # Непосредственно генерируем фильтр
    alphas = []
    betas = []
    our_filter = 1
      a_constants = [1, -2 * np.cos(((2 * order_number + filter_order - 1)/(2 * filter_order)) * np.pi), 1]

    current_betta = [1, 0, -2, 0, 1]

      current_alpha = [gamma**2 * a_constants[0] + gamma * a_constants[1] + a_constants[2],
                    -4 * gamma**2 * zeta * a_constants[0] - 2 * gamma * zeta * a_constants[1],
```

```
                    4 * gamma**2 * zeta**2 * a_constants[0] + 2 * gamma**2 *
a_constants[0] - 2 * a_constants[2],
                   -4 * gamma**2 * zeta * a_constants[0] + 2 * gamma * zeta *
a_constants[1],
          gamma**2 * a_constants[0] - gamma * a_constants[1] + a_constants[2]]

    our_filter = our_filter * ((current_betta[0] * z**4 + current_betta[1] * z**3 +
current_betta[2] * z**2 + current_betta[3] * z + current_betta[4])/ \
      (current_alpha[0] * z**4 + current_alpha[1] * z**3 + current_alpha[2] * z**2
+ current_alpha[3] * z + current_alpha[4]))
   alphas = np.array(current_alpha)/current_alpha[0]
   betas = np.array(current_betta)/current_alpha[0]

   return np.array(our_filter), np.array(alphas), np.array(betas)


def recursive_transfer_function(z, fn=0, fv=1):
   """
     Генерирует теоритически заданный аналоговый прототип ПФ фильтра
Баттерворта
   соответственно входным данным

   :z: оператор передаточной функции
   :fn: Нижняя граница частоты среза фильтра
   :fv: Верхняя граница частоты среза фильтра
   :returns: расчитанную передаточную функцию
   """
   import numpy as np

   fn = fn * 2 * np.pi
   fv = fv * 2 * np.pi

   # Вводим константы
   gamma = 1/np.tan((delta_T/2) * (fv - fn))
   zeta = np.cos((delta_T/2) * (fv + fn))/np.cos((delta_T/2) * (fv - fn))

   # Осуществляем замену переменной p из ФНЧ в ПФ
   changed_p = gamma * (((z**2) - 2 * zeta * z + 1)/((z**2) - 1))

   # Записываем передаточную функцию фильтра Баттерворта 2-ого порядка
   calculated_transfer_function = 1/((changed_p**2) + 1.41421 * changed_p + 1)

   return calculated_transfer_function
```

```
def main():
    import numpy as np
    import matplotlib.pyplot as plt

    # Вычисляем частоту
    k = np.linspace(0, q, q)
    frequency = k/(q * delta_T)

    # Формируем массив коэффициентов
    coefficient = []
    for frequency_number in frequency:
        coefficient.append(complex(0, 2 * np.pi * frequency_number * delta_T))

    # Расчёт теоритического рекурсивного фильтра по передаточной функции
    filter_transfer_function = recursive_transfer_function(np.exp(np.array(coefficient)), fv=410, fn=390)

    # Расчёт рекурсивного фильтра
    recursive_filter, A, B = generate_recursive_filter(np.exp(np.array(coefficient)), fv=410, fn=390)
    fn_div_10 = 20 * np.log10(abs(generate_recursive_filter(np.exp(complex(0, 2 * np.pi * 39 * delta_T)), fv=410, fn=390)[0]))
    fv_mul_10 = 20 * np.log10(abs(generate_recursive_filter(np.exp(complex(0, 2 * np.pi * 4100 * delta_T)), fv=410, fn=390)[0]))

    # Вывод коэффициентов рекурсивного фильтра
    print("Альфа коэффициенты")
    for coef_number in range(0, len(A)):
        print(f'{coef_number + 1}-ый: {A[coef_number]:.64}')
    print('Бета коэффициенты')
    for coef_number in range(0, len(B)):
        print(f'{coef_number + 1}-ый: {B[coef_number]:.64}')

    # Сравниваем теоритическое АЧХ рекурсивного фильтра и АЧХ рассчитаного фльтра
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.plot(frequency[:int(q/2)], 20 * np.log10(abs(filter_transfer_function))[:int(q/2)], label='АЧХ фильтра рассчитанного по аналоговому прототипу')
    ax.plot(frequency[:int(q/2)], 20 * np.log10(abs(recursive_filter))[:int(q/2)], label='АЧХ фильтра рассчитаного по таблице')
    ax.plot(frequency[:int(q/2)], -60 * np.ones(int(q/2)), '--', label='Граница коэффициента подавления -60 дБ')
    ax.text(0, 0.5, f'Глубина затухания при 39 Гц = {round(fn_div_10)}',
```

```
transform=ax.transAxes, fontsize=14)
        ax.text(0, 0.4, f'Глубина затухания при 4100 Гц = {round(fv_mul_10)}',
transform=ax.transAxes, fontsize=14)
    ax.set_xlabel('Частота, Гц')
    ax.set_ylabel('Коэффициент подавления, дБ')
    ax.set_xscale('log')
    ax.legend()

    plt.show()


if __name__ == "__main__":
    main()
```