## Федеральное агентство по образованию

#### Государственное образовательное учреждение высшего профессионального образования

**«Московский государственный технический университет имени Н.Э.Баумана»**

****

**Домашнее задание**

по курсу: «Цифровая обработка оптических сигналов»

Вариант 3

Студент ( *Метелкина А.А.* ) Группа *РЛ2-102*

(фамилия, инициалы) (индекс)

Преподаватель ( *Бокшанский В.Б )*

(фамилия, инициалы)

Москва, 2017

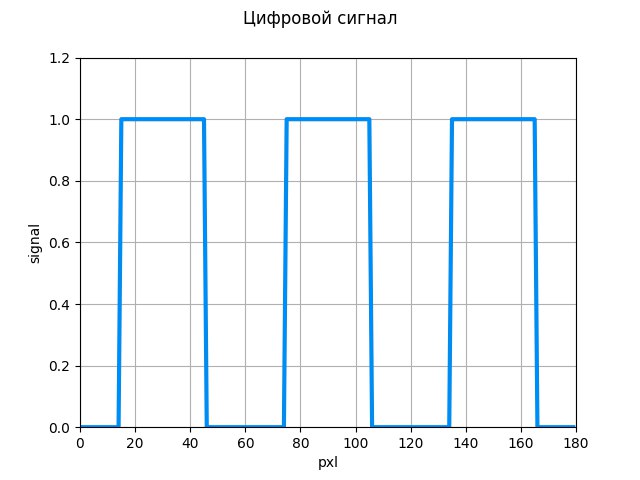
**Задание.**

Синтезировать одномерный симметричный цифровой фильтр, соответствующая аналоговая характеристика которого представлена ниже. Осуществить цифровую фильтрацию заданного в виде сечения тестового изображения, зарегистрированного матричным фотоприемником с числом элементов 180, размер одного элемента 2х2 мкм. Для этого:

1. Рассчитать коэффициенты фильтра *h(i,j);*
2. Построить характеристику фильтра в зависимости от нормированной частоты.
3. Осуществить НЧ-фильтрацию заданного изображения с помощью программы MathLab или MathCAD.
4. Синтезировать двумерный октально-симметричный фильтр, использую полученные коэффициенты одномерного фильтра. Провести проверку корректности синтезированного фильтра.
5. Осуществить двумерную цифровую фильтрацию полутонового изображения в соответствие с вариантом.

Порядок фильтра N=3

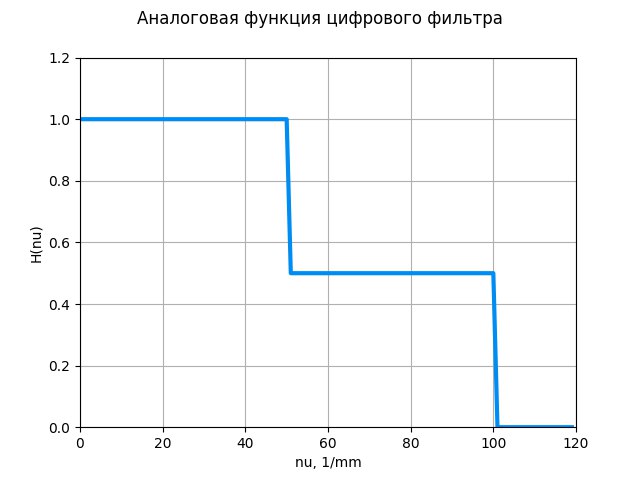
Форма цифрового сигнала



*Рисунок 1.* Исходный цифровой сигнал

**Решение:**

Аналоговая функция цифрового фильтра:

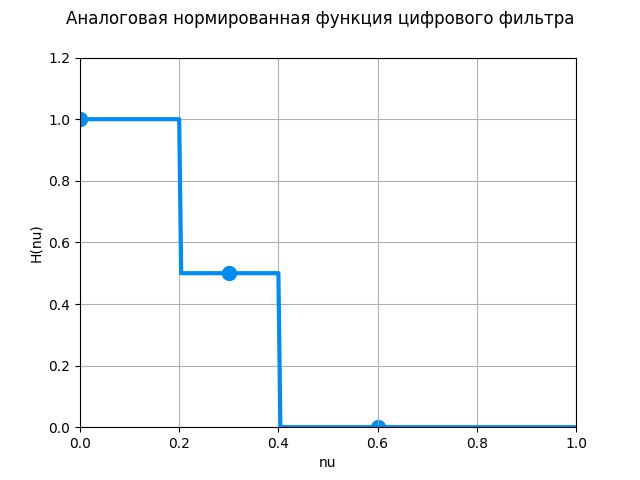


*Рисунок 2.* Аналоговая функция цифрового фильтра

Аналоговая функция цифрового фильтра для нормированной частоты:

где  ­– частота Найквиста, которая для матричного приемника с размером пикселя 2×2 мкм равна:

Тогда:



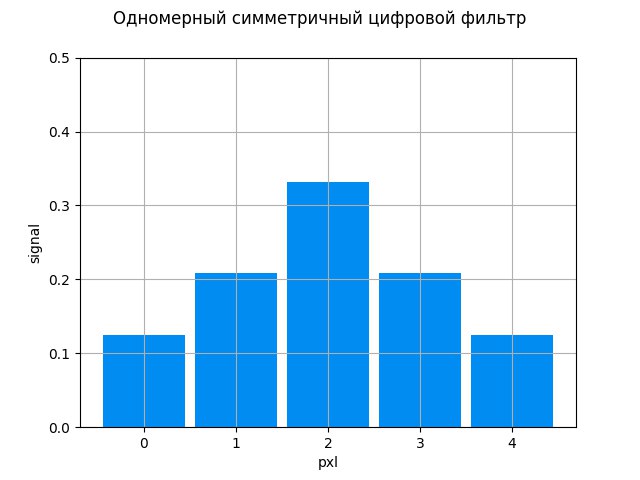
*Рисунок 3.* Аналоговая функция цифрового фильтра для нормированной частоты

Для создания фильтра 4го порядка, необходимо взять четыре точки отсчёта на приведённой характеристике:

Для определения коэффициентов фильтра составим систему уравнений в матричной форме:

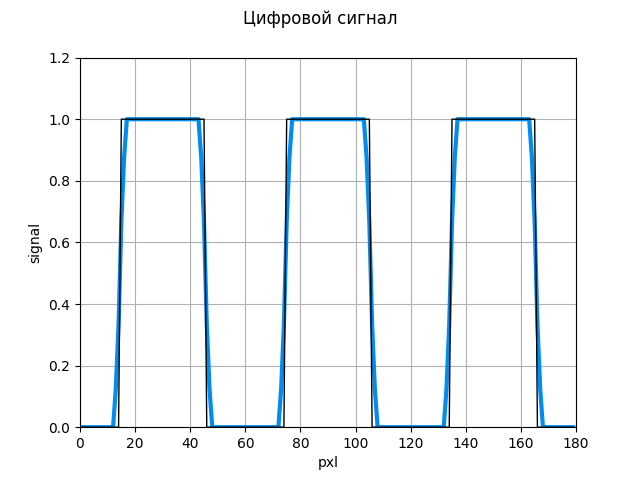
В итоге, получили следующие значения коэффициентов фильтра:

Вид симметричного одномерного фильтра:

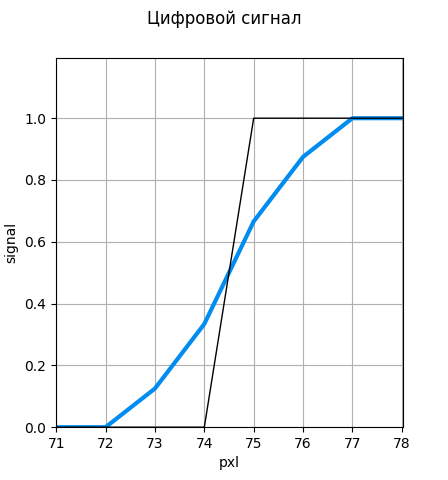


*Рисунок 4.* Цифровой одномерный симметричный фильтр

Применим синтезированный фильтр к исходному сигналу, представленному на Рисунке 1. Результат фильтрации представлен на Рисунке 5.



*Рисунок 5.* Вид сигнала после фильтрации



*Рисунок 5.* Увеличенный фрагмент сигнала после фильтрации

Вид симметричного двумерного фильтра с октальной симметрией:

Т.к. сумма коэффициентов фильтра ≠ 0, необходимо провести нормировку для выполнения условия сохранения энергии

Исходное изображение имеет размер . Т. К. размер пикселя равен , изображение будет занимать .

Исходное и отфильтрованное изображения представлены на Рисунке 6, 7 соответственно.

|  |  |
| --- | --- |
| C:\Users\metel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\im_2.png  Рисунок 6. Исходное полутоновое изображение | Рисунок 7. Полутоновое изображение после фильтрации |
| Рисунок 8. Фрагмент отфильтрованного полутоновое изображения |  |

**Исходный код (Python)**

import numpy as np

import matplotlib.pyplot as plt

M = 180 # Размер матричного ПИ

a = 2 # мкм, Размер элемента ПИ

n = 3 # Порядокфильтра

""" Аналоговая функция фильтра"""

def H(X, coef=1):

y = []

for x in X:

if np.abs(x) <= 50 / coef:

y.append(1.0)

elif np.abs(x) > 100 / coef:

y.append(0.0)

else:

y.append(0.5)

return y

grid = np.arange(0, 120)

H1 = H(grid)

plt.figure()

plt.plot(grid, H1, linewidth=3, color='#008cf0')

plt.suptitle('Аналоговая функция цифрового фильтра')

plt.xlabel('nu, 1/mm')

plt.ylabel('H(nu)')

plt.ylim([0, 1.2])

plt.xlim([0, 120])

plt.grid()

plt.show()

""" Частота Найквиста"""

nu\_N = 1 / (2 \* a / 1000)

print('Частота Найквиста = %0.2f 1/mm' % nu\_N)

""" Нормированная аналоговая функция фильтра"""

grid\_norm = np.arange(0, 1, 1 / nu\_N)

H1 = H(grid\_norm, nu\_N)

""" Дискретные значения"""

grid\_d = np.arange(0, 0.7, 0.3)

H\_d = H(grid\_d, nu\_N)

print('Дискретные отсчеты: \n', grid\_d, '\n Матрица разложения: \n', H\_d)

plt.figure()

plt.scatter(grid\_d, H\_d, linewidth=5, color='#008cf0')

plt.plot(grid\_norm, H1, linewidth=3, color='#008cf0')

plt.suptitle('Аналоговая нормированная функция цифрового фильтра')

plt.xlim([0, 1])

plt.ylim([0, 1.2])

plt.xlabel('nu')

plt.ylabel('H(nu)')

plt.grid()

plt.show()

m = np.ones([3, 3])

for i in range(3):

for j in range(1, 3):

m[i, j] = 2 \* np.cos(j \* np.pi \* grid\_d[i])

print(m)

""" Коэффициенты фильтра"""

h = np.linalg.solve(m, H\_d)

checked\_H = np.dot(m, np.reshape(h, [-1, 1]))

""" Формирование одномерного массива"""

h1d = np.concatenate((np.flip(h[1:], 0), h))

grid\_d2 = np.concatenate((-np.flip(grid\_d[1:], 0), grid\_d))

plt.figure()

plt.bar(np.arange(5), h1d, width=0.9, color='#008cf0')

plt.suptitle('Одномерный симметричный цифровой фильтр')

plt.ylim([0, 0.5])

plt.xlabel('pxl')

plt.ylabel('signal')

plt.grid()

plt.show()

""" Формирование функции одномерного цифрового сигнала """

def rect(x, coef=1):

y = []

for i in x:

if np.abs(i - len(x) / 2) <= len(x) / 4 / coef:

y.append(1.0)

else:

y.append(0.0)

return np.array(y)

grid\_signal = np.arange(int(M / n))

signal = np.concatenate((rect(grid\_signal), rect(grid\_signal), rect(grid\_signal)))

plt.figure()

plt.plot(signal, linewidth=3, color='#008cf0')

plt.suptitle('Цифровой сигнал')

plt.xlim([0, M])

plt.ylim([0, 1.2])

plt.xlabel('pxl')

plt.ylabel('signal')

plt.grid()

plt.show()

def my\_conv(f, kernel):

pad = int(len(kernel)/2-1/2)

if len(f.shape)==1:

filt = np.zeros(f.size)

f = np.pad(f, [pad, pad], mode='edge')

for i in range(pad, len(f) - pad):

for j in range(len(kernel)):

filt[i - pad] += kernel[j] \* f[i - int(len(kernel) / 2 - 1 / 2) + j]

else:

filt = np.zeros(f.shape)

f = np.pad(f, ([pad, pad], [pad, pad]), mode='edge')

for i in range(pad, len(f) - pad):

for i2 in range(pad, len(f) - pad):

for j in range(len(kernel)):

for j2 in range(len(kernel)):

filt[i-pad, i2-pad] += kernel[j, j2] \* f[i - int(len(kernel) / 2 - 1 / 2)+j,

i2 - int(len(kernel) / 2 - 1 / 2) + j2]

return filt

s\_f = my\_conv(signal, h1d)

""" Фильтрация одномерного сигнала фильтром"""

plt.figure()

# plt.plot(signal\_f, linewidth=3, color='#008cf0')

plt.plot(s\_f, linewidth=3, color='#008cf0')

plt.plot(signal, linewidth=1, color='#000000')

plt.suptitle('Цифровой сигнал')

plt.xlim([0, M])

plt.ylim([0, 1.2])

plt.xlabel('pxl')

plt.ylabel('signal')

plt.grid()

plt.show()

""" Формирование функции двумерного цифрового сигнала ""

def img(im\_size=180):

n\_p = int(im\_size / a)

y = np.zeros([M, M])

y[int(M / 2 - n\_p / 2):int(M / 2 + n\_p / 2), int(M / 2 - 0.4 \* n\_p / 2):int(M / 2 + 0.4 \* n\_p / 2)] = 1

return

def add\_frame(im, frame\_s=180):

image = np.copy(im)

n\_p = frame\_s / 2

image[int(M / 2 - n\_p / 2), int(M / 2 - n\_p / 2):int(M / 2 + n\_p / 2)]= 1

image[int(M / 2 + n\_p / 2), int(M / 2 - n\_p / 2):int(M / 2 + n\_p / 2)]= 1

image[int(M / 2 - n\_p / 2):int(M / 2 + n\_p / 2), int(M / 2 + n\_p / 2)]= 1

image[int(M / 2 - n\_p / 2):int(M / 2 + n\_p / 2), int(M / 2 - n\_p / 2)]= 1

return image

im = img(im\_size=360)

plt.figure()

plt.imshow(im, cmap='gray')

plt.suptitle('Исходное изображение')

plt.show()

""" Формирование двумерного симметричного фильтра ""

def make\_sym(a, dtype=float):

s = len(a) \* 2 - 1

m = np.zeros([s, s], dtype=dtype)

m.fill(a[-1])

for i in range(1, len(a)):

m[i:s - i, i:s - i].fill(a[-(i + 1)])

return m

h2d = make\_sym(h)

plt.figure()

plt.plot(h2d[2])

plt.suptitle('Изображение после фильтрации')

plt.show()

h2d = h2d / np.sum(h2d) # Нормировка

print(h2d)

""" Фильтрация одномерного сигнала фильтром"""

im\_f = convolve(im, h2d)

im2 = my\_conv(im,h2d)

plt.figure()

plt.imshow(im2, cmap='gray')

plt.suptitle('Изображение после фильтрации')

plt.show()