ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | Т.А. Суетина |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| Сжатие изображения классическим алгоритмом JPEG |
| по курсу: Техника аудиовизуальных средств информации |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4329 |  |  |  | Д.С. Шаповалова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2025

# **1. Цель работы:**

Получить теоретические знания и практические навыки по сжатию изображения классическим алгоритмом JPEG.

# **2. Задание:**

Выполнить сжатие изображения 16 на 16 точек в цветовом пространстве RGB при помощи классического алгоритма JPEG.

Исходное изображение представлено на рисунке 1.1:



Рисунок 1.1 – Исходное изображение

# **3. Ход работы:**

1. Объём входного файла = 16\*16\*3\*8 = 6144 бит.

2. Преобразование цветового пространство исходного изображения из RGB в YCbCr. Для этого используются функции в Matlab.

Полученные матрицы представлены на рисунках 2.1 – 2.3:

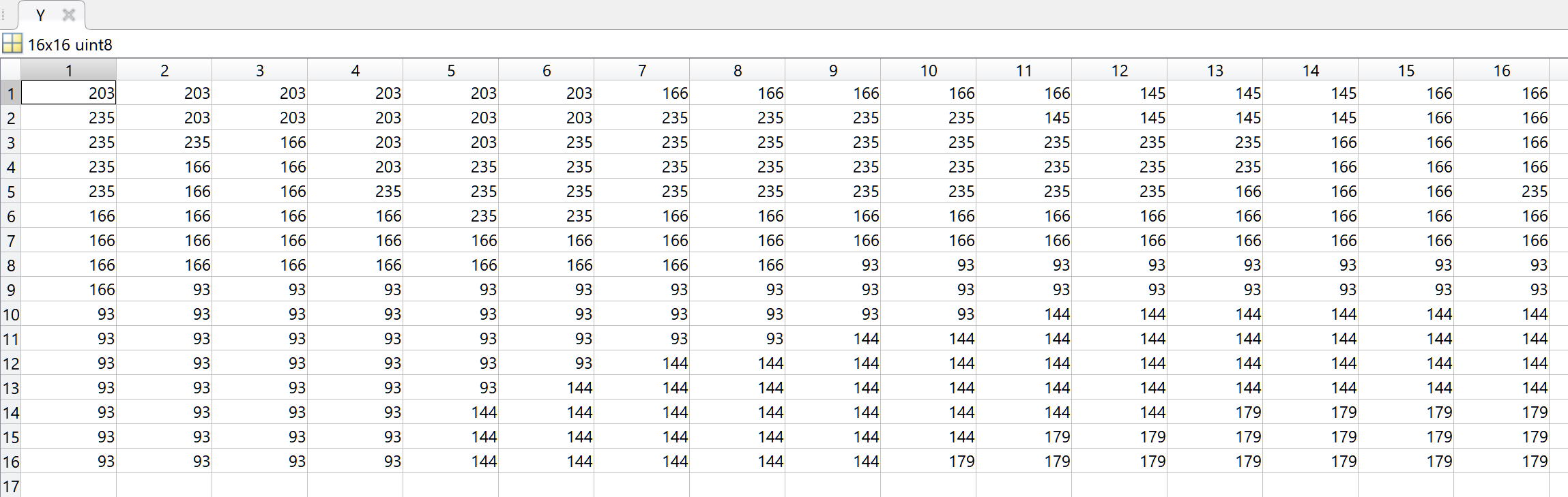


Рисунок 2.1 – Матрица яркостного компонента Y

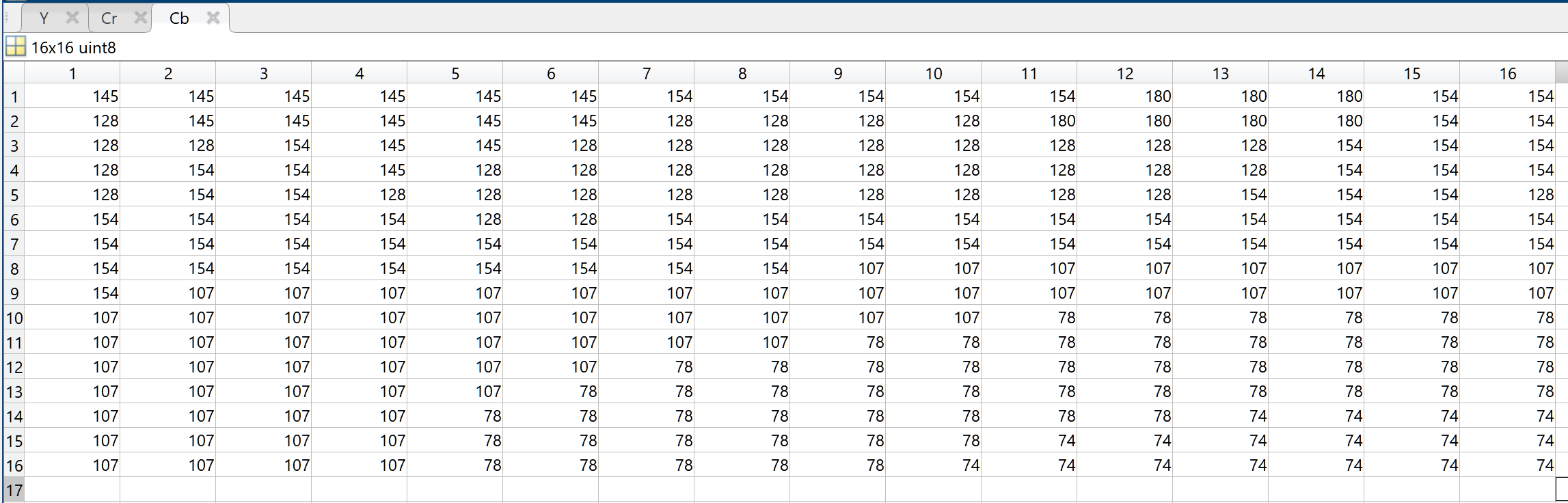


Рисунок 2.2 – Матрица цветоразностного компонента Cb

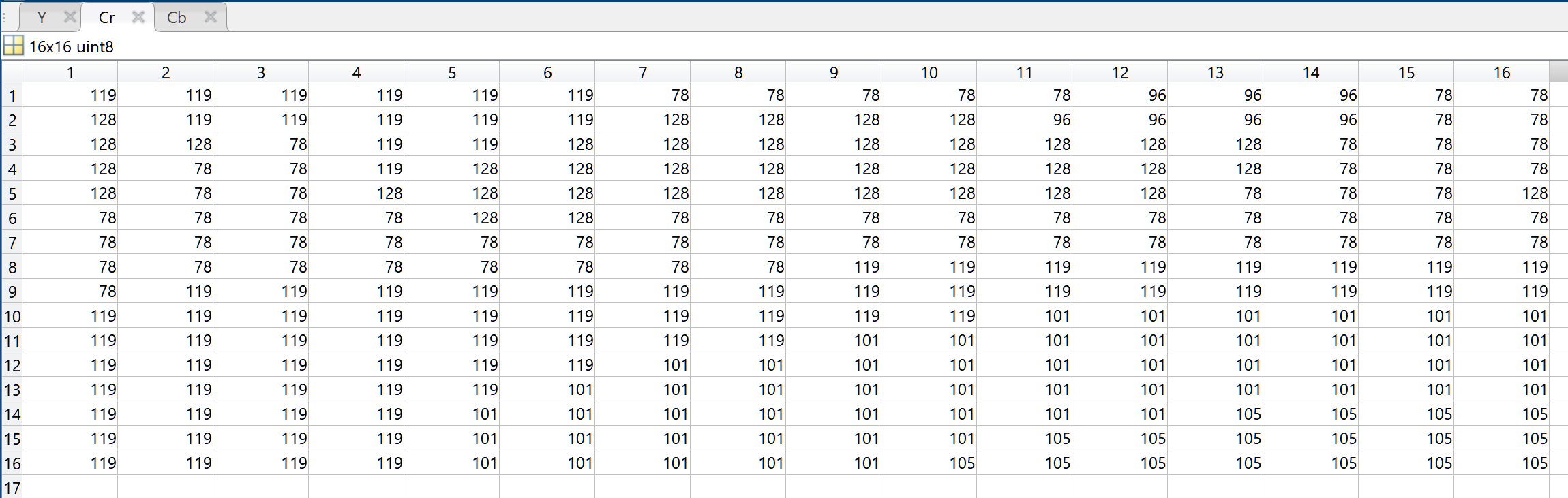


Рисунок 2.3 – Матрица цветоразностного компонента Cr

3. Прореживание.

Для компонентов Cr и Cb для каждого блока 2х2 находится и записывается среднее значение блока.

Матрица яркостного компонента разбивается на матрицы размерностью 8х8 пикселей.

Полученные матрицы представлены на рисунках 3.1-3.3.4:

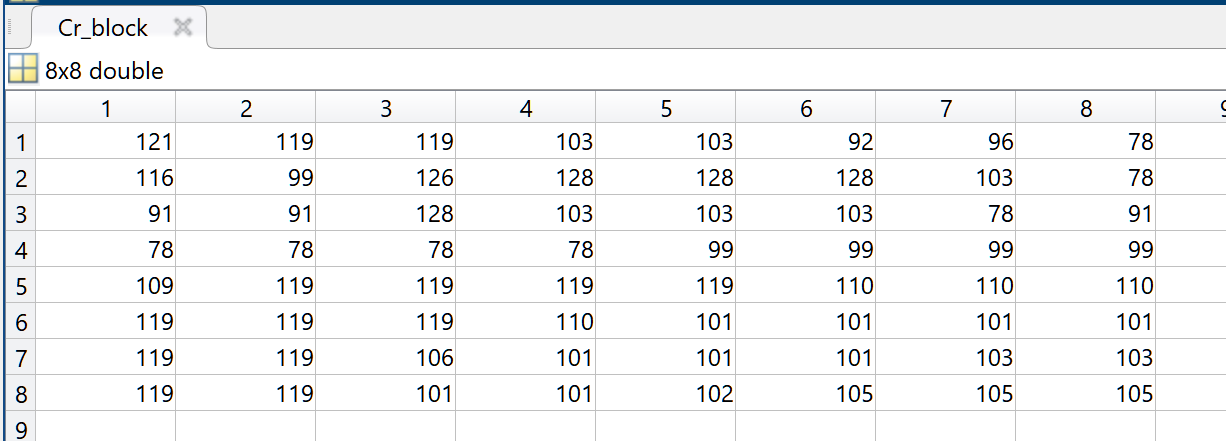


Рисунок 3.1 – Усреднённая матрица цветоразностного компонента Cr

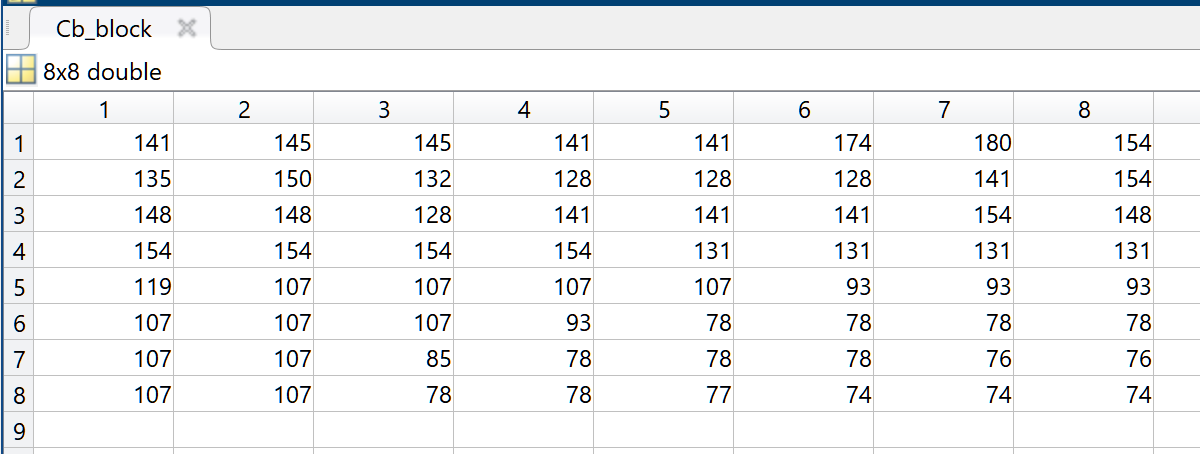


Рисунок 3.2 – Усреднённая матрица цветоразностного компонента Cb

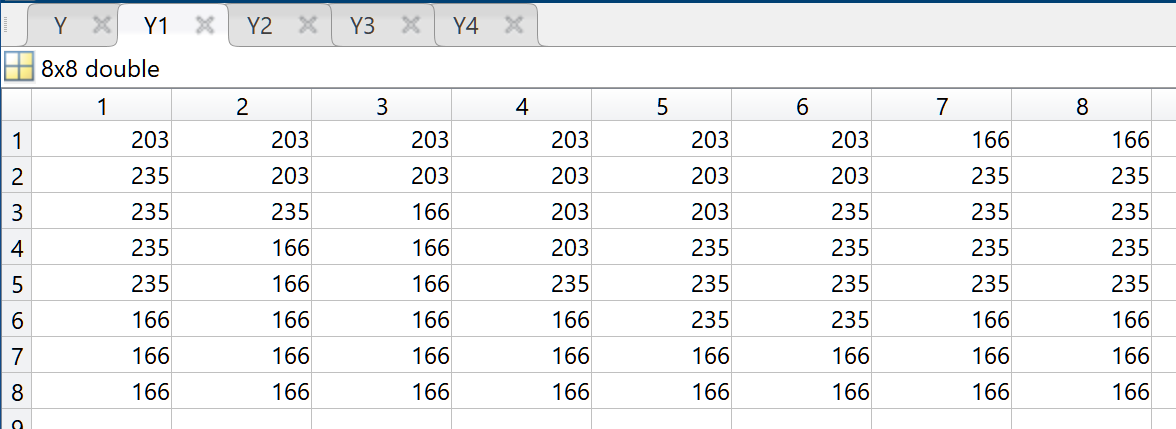


Рисунок 3.3.1 – Часть 1 матрицы яркостного компонента

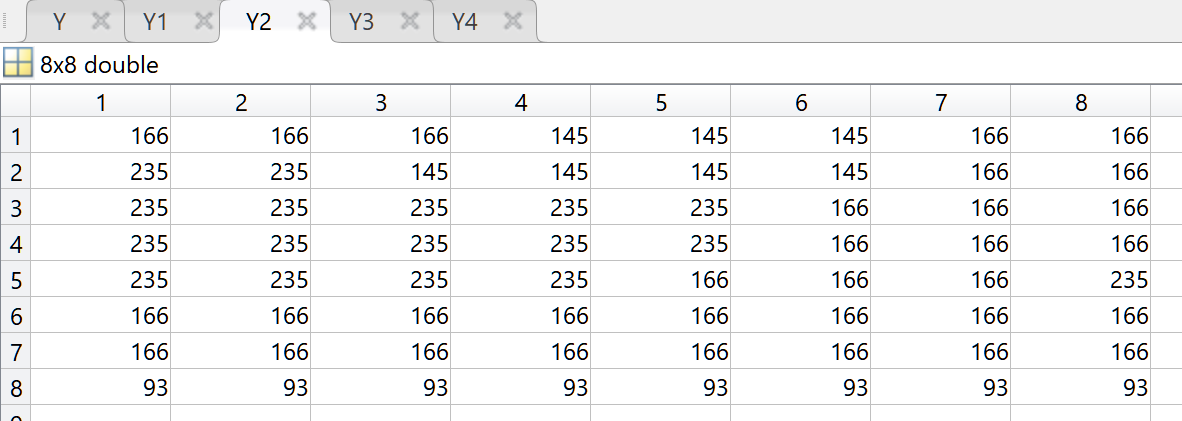


Рисунок 3.3.2 – Часть 2 матрицы яркостного компонента

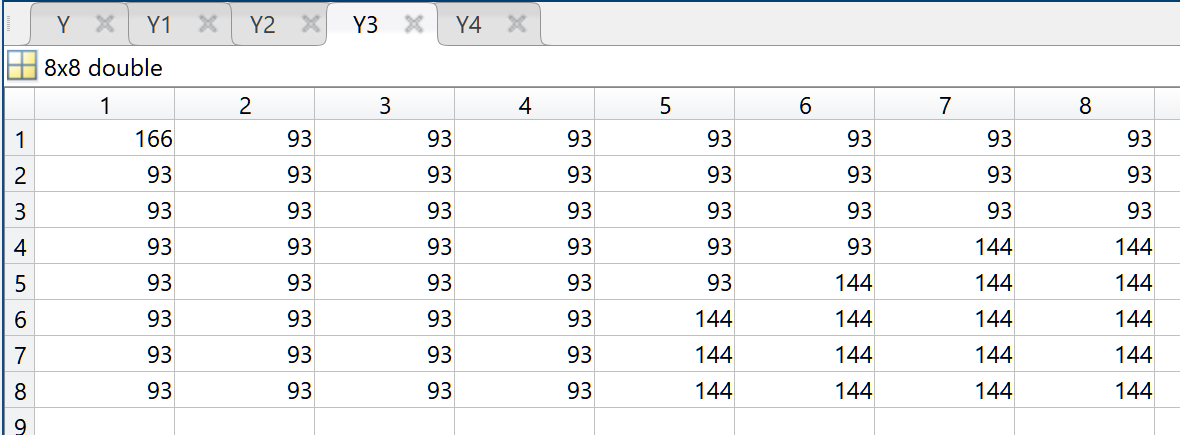


Рисунок 3.3.3 – Часть 3 матрицы яркостного компонента

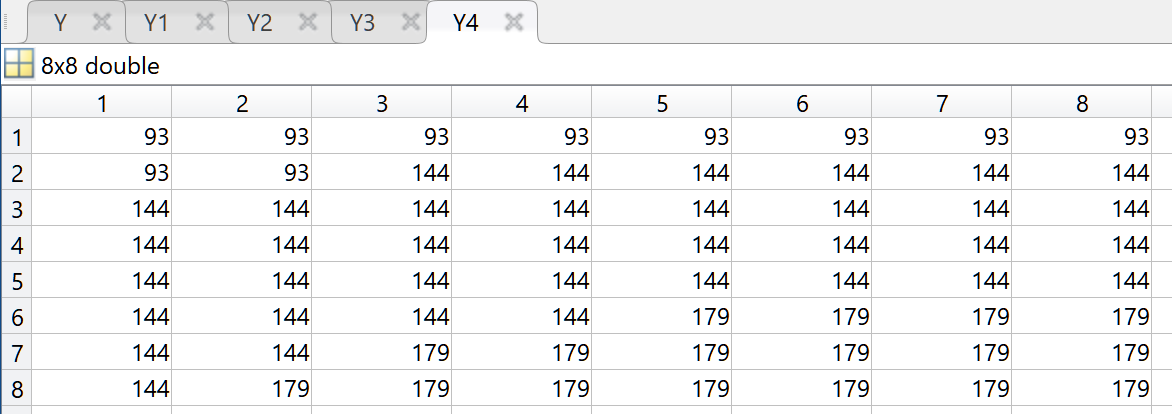


Рисунок 3.3.4 – Часть 4 матрицы яркостного компонента

4. Двумерное косинусное преобразование (ДКП)

Использована функция Matlab – M=dctmtx(8), произведено умножение каждой полученной матрицы 8х8 на матрицу коэффициентов М по формуле:

,

где V – рабочие матрицы.

Полученные матрицы представлены на рисунках 4.1-4.3.4:

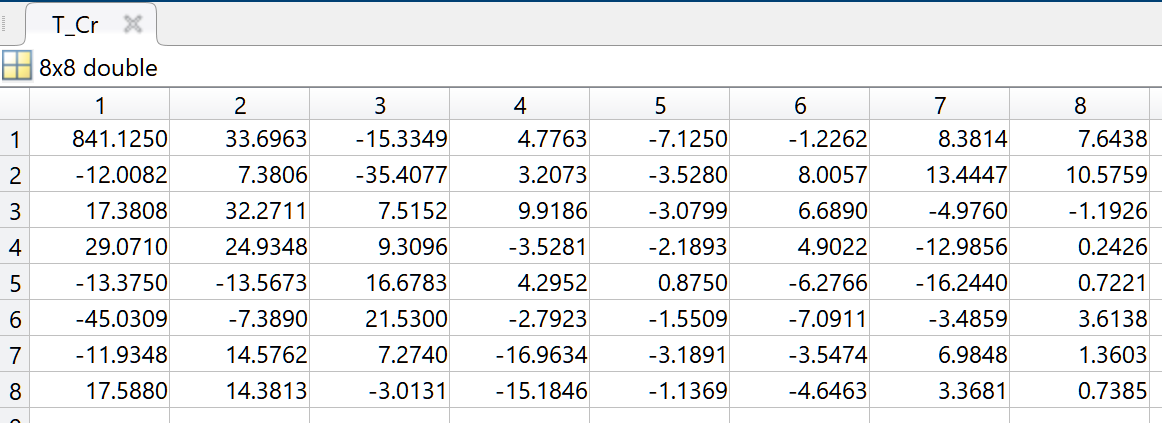


Рисунок 4.1 – Результат ДКП для матрицы цветоразностного компонента Cr

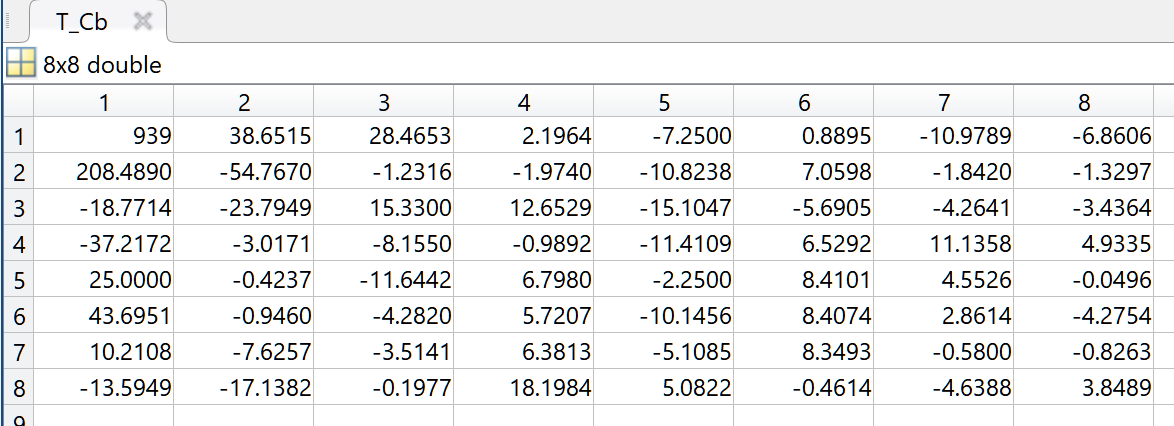


Рисунок 4.2 – Результат ДКП для матрицы цветоразностного компонента Cb

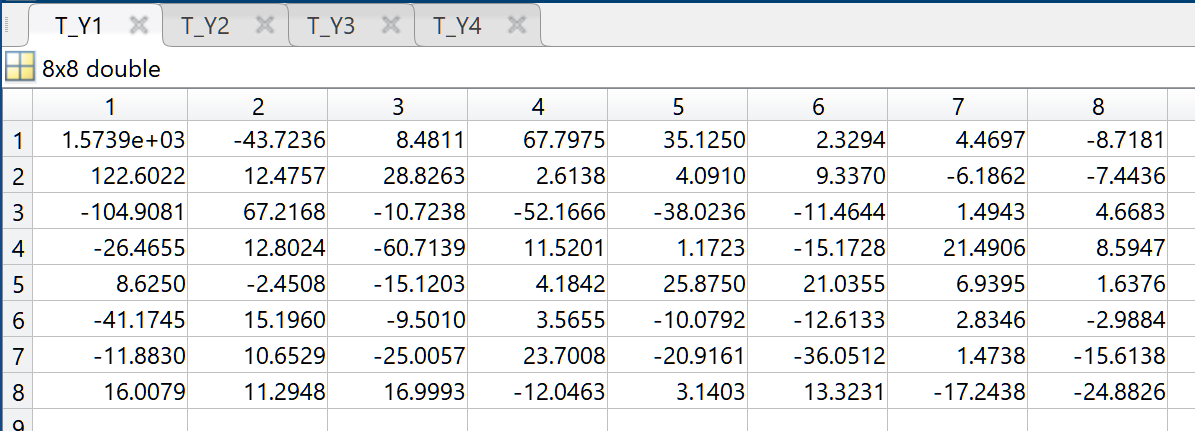


Рисунок 4.3.1 – Результат ДКП для части 1 матрицы яркостного компонента Y

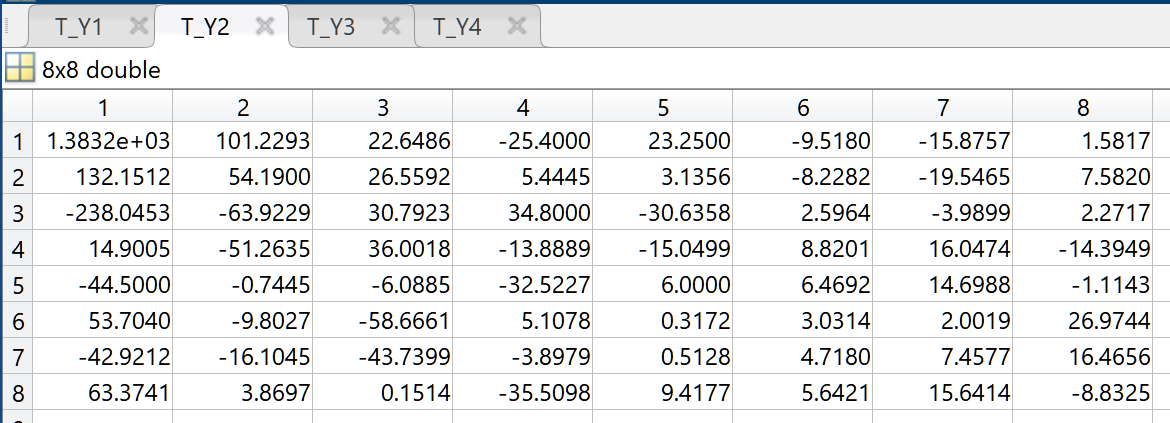


Рисунок 4.3.2 – Результат ДКП для части 2 матрицы яркостного компонента Y

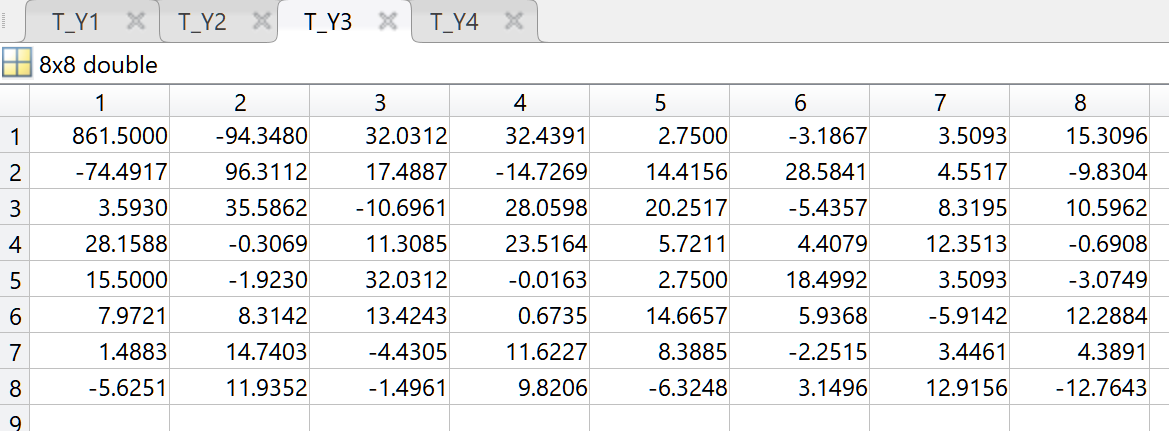


Рисунок 4.3.3 – Результат ДКП для части 3 матрицы яркостного компонента Y

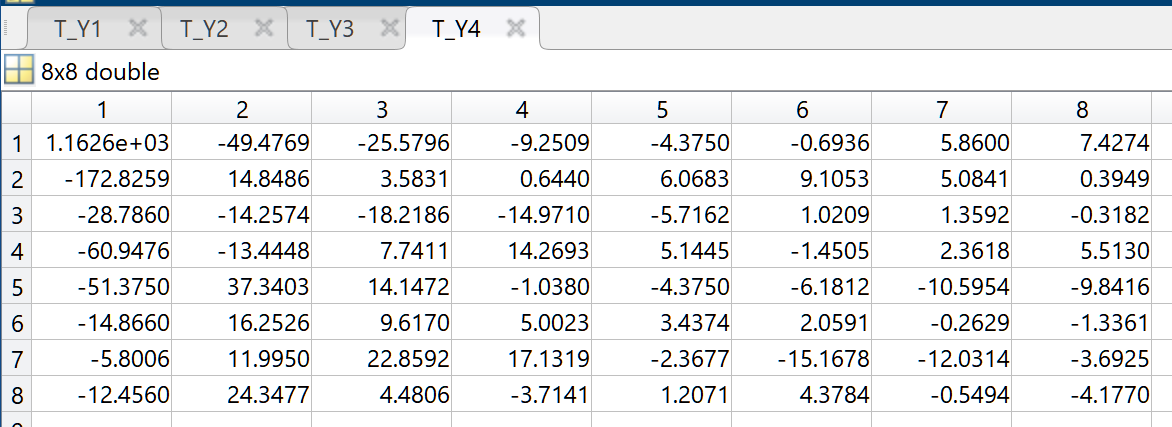


Рисунок 4.3.4 – Результат ДКП для части 4 матрицы яркостного компонента Y

5. Квантование

Поэлементно делим матрицы яркостного и цветоразностного компонентов на матрицы коэффициентов квантования для соответствующих компонентов.

Полученные матрицы представлены на рисунках 5.1-5.3.4:

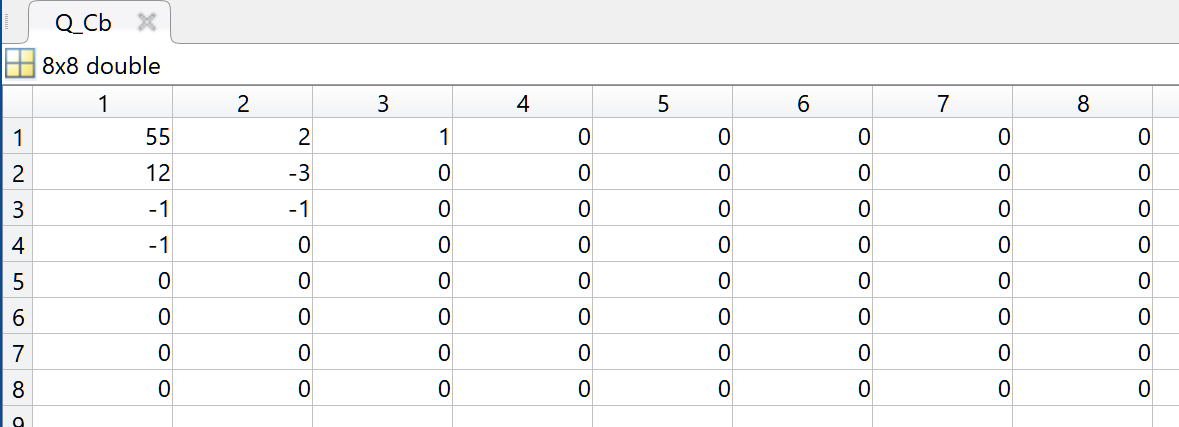


Рисунок 5.1 – Результат квантования матрицы цветоразностного компонента Cb

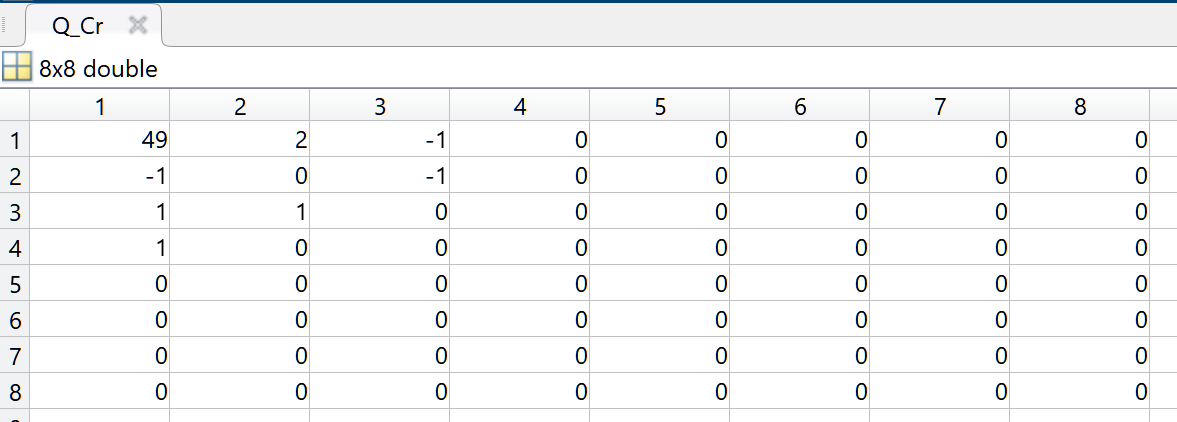


Рисунок 5.2 – Результат квантования матрицы цветоразностного компонента Cr

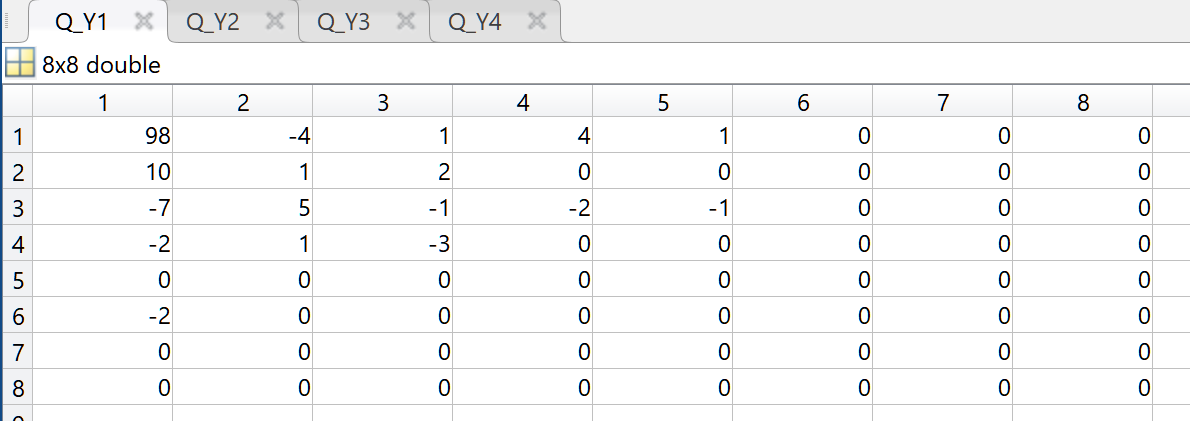


Рисунок 5.3.1 – Результат квантования матрицы 1 яркостного компонента Y

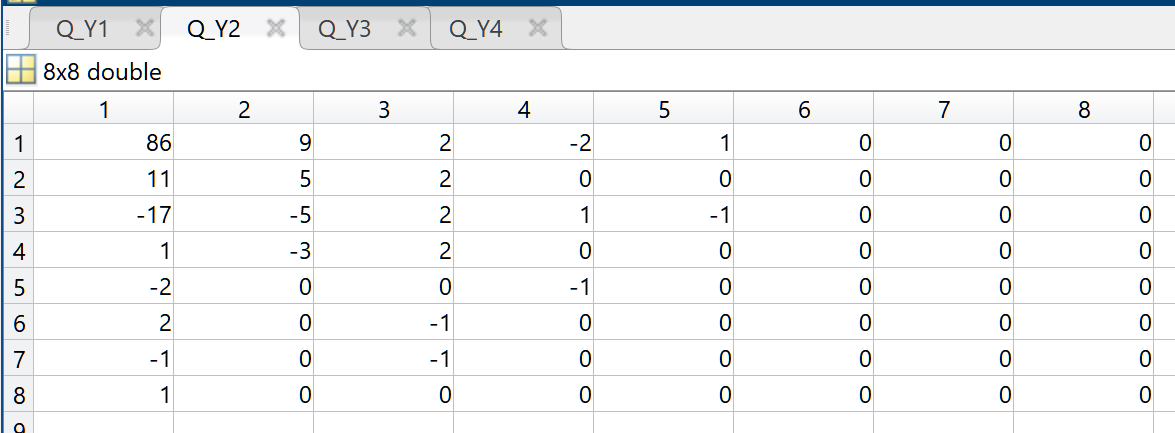


Рисунок 5.3.2 – Результат квантования матрицы 2 яркостного компонента Y

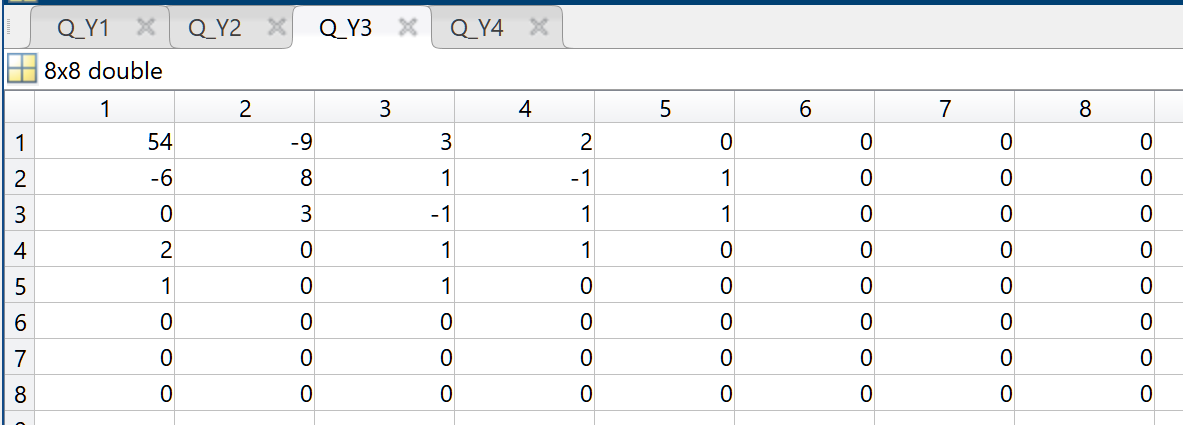


Рисунок 5.3.3 – Результат квантования матрицы 3 яркостного компонента Y

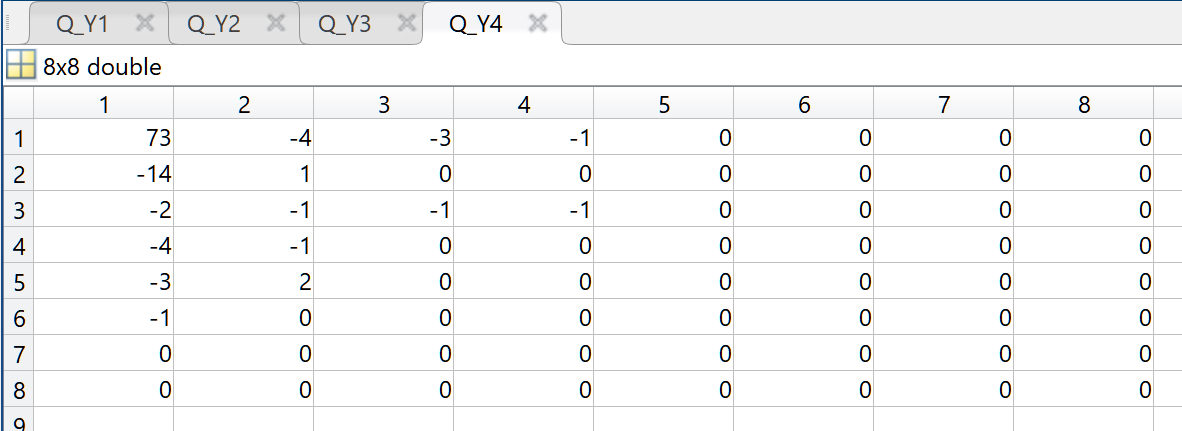


Рисунок 5.3.4 – Результат квантования матрицы 4 яркостного компонента Y

6. Алгоритм диагонального сканирования (зиг-заг обход)

Представление полученных матриц как векторов:

Cr = [49, 2, -1, 1, 0, -1, 0, -1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Cb = [55, 2, 12, -1, -3, 1, 0, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Y1 = [98, -4, 10, -7, 1, 1, 4, 2, 5, -2, 0, 1, -1, 0, 1, 0, 0, -2, -3, 0, -2, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Y2 = [86, 9, 11, -17, 5, 2, -2, 2, -5, 1, -2, -3, 2, 0, 1, 0, 0, 1, 2, 0, 2, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, -1, 0, 1, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Y3 = [54, -9, -6, 0, 8, 3, 2, 1, 3, 2, 1, 0, -1, -1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Y4 = [73, -4, -14, -2, 1, -3, -1, 0, -1, -4, -3, -1, -1, 0, 0, 0, 0, -1, 0, 2, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

7. Сворачиваем повторяющиеся значения по алгоритму RLE в формате (количество повторений, повторяющееся значение):

Cr = [49, 2, -1, 1, 0, -1, 0, -1, (2, 1), (55, 0), FF]

Cb = [55, 2, 12, -1, -3, 1, (2, 0), (2, -1), (54, 0), FF]

Y1 = [98, -4, 10, -7, (2, 1), 4, 2, 5, -2, 0, 1, -1, 0, 1, (2, 0), -2, -3, 0, -2, (4, 0), -1, (38, 0), FF]

Y2 = [86, 9, 11, -17, 5, 2, -2, 2, -5, 1, -2, -3, 2, 0, 1, (2,0), 1, 2, 0, 2, -1, (3, 0), -1, (6, 0), (2, -1), 0, 1, 0, -1, (22, 0), FF]

Y3 = [54, -9, -6, 0, 8, 3, 2, 1, 3, 2, 1, 0, (2, -1), (2, 0), (3, 1), (4, 0), (3, 1), (38, 0), FF]

Y4 = [73, -4, -14, -2, 1, -3, -1, 0, -1, -4, -3, (2, -1), (4, 0), -1, 0, 2, -1, (43, 0), FF]

8. Кодирование по Хаффману

Осуществляется в соответствии с таблицей:

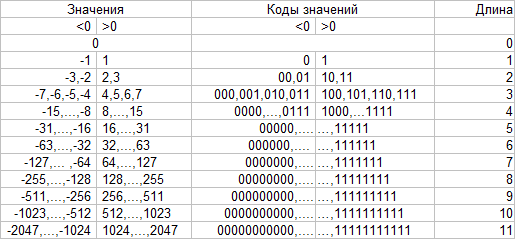


Рисунок 6 – Таблица кодов по Хаффману

После кодирования получаем такие значения:

Cb = 10 + 6 + 8 + 5 + 6 + 5 + 5 + 5 + 4 = 54 бит

Cr = 10 + 6 + 5 + 5 + 5 + 5 + 5 + 5 + 4 = 50 бит

Y1 = 11 + 7 + 8 + 7 + 5 + 5 + 7 + 6 + 7 + 6 + 5 + 5 + 5 + 6 + 6 + 6 + 5 + 4 = 111 бит

Y2 = 11 + 8 + 8 + 9 + 7 + 6 + 6 + 6 + 7 + 5 + 6 + 6 + 6 + 5 + 5 + 6 + 6 + 5 + 5 + 5 + 4 = 147 бит

Y3 = 10 + 8 + 7 + 8 + 6 + 6 + 5 + 6 + 6 + 5 + 5 + 5 + 5 + 5 + 5 + 4 = 111 бит

Y4 = 11 + 7 + 8 + 6 + 5 + 6 + 5 + 5 + 7 + 6 + 5 + 5 + 5 + 0 + 6 + 5 + 4 = 96 бит

Размер сжатого изображения:

V = 54 + 50 + 111 + 147 + 111 + 96 = 569 (бит)

Коэффициент сжатия:

# **4. Вывод:**

В ходе лабораторной работы был подробно рассмотрен классический алгоритм сжатия JPEG. Было выполнено пошаговое сжатие изображения 16х16 пикселей, с глубиной цвета 3 бита и цветовым пространством RGB.

Объём исходного файла составлял 6144 бита, после алгоритма JPEG составляет 569 бит, что даёт нам коэффициент сжатия в 10,8 раз.

Были выполнены следующие шаги:

1. Преобразование из RGB в YCbCr – матрицы яркостного и цветоразностных компонентов.
2. Прореживание на блоки 8х8 для компонента Y и усреднение в 2 раза для компонентов Cb и Cr.
3. Дискретно косинусное преобразование (ДКП 2) для разделения низких частот и высоких – переводит матрицу значений пикселей 8х8 в матрицу коэффициентов ДКП 8х8, представляющих частоты изображения. Большая часть энергии (визуально значимой информации) сосредоточена в левом верхнем углу матрицы коэффициентов, а наименьшая — в правом нижнем.
4. Квантование – с использованием стандартных матриц квантовани – обнуляет часть высокочастотных коэффициентов.
5. Зиг-заг сканирование – эффективное расположение коэффициентов в порядке возрастания частоты.
6. RLE-кодирование – запись повторяющихся значений более кратко.
7. Кодирование по Хаффману – использование эффективных кодовых слов для представления информации.

Подводя итог работы, можно сказать, что самый ценный этап в сжатии JPEG – прореживание и квантование – моменты, где теряется информация, детали изображения, высокие частоты, незаметные глазу человека.

# ПРИЛОЖЕНИЕ А

Листинг Программы

rgb=imread ('task.jpg');

ycbcr=rgb2ycbcr(rgb);

%Преобразование в YcRcB

Y = ycbcr(:,:,1);

Cb = ycbcr(:,:,2);

Cr = ycbcr(:,:,3);

%Разделение на 8х8

Cb\_block = round(blockproc(Cb, [2 2], @(block) mean(block.data(:))));

Cr\_block = round(blockproc(Cr, [2 2], @(block) mean(block.data(:))));

Y1 = double(Y(1:8, 1:8));

Y2 = double(Y(1:8, 9:16));

Y3 = double(Y(9:16, 1:8));

Y4 = double(Y(9:16, 9:16));

%ДКП

M = dctmtx(8);

T\_Y1 = M \* Y1 \* M';

T\_Y2 = M \* Y2 \* M';

T\_Y3 = M \* Y3 \* M';

T\_Y4 = M \* Y4 \* M';

T\_Cb = M \* Cb\_block \* M';

T\_Cr = M \* Cr\_block \* M';

%Квантование

QY = [16 11 10 16 24 40 51 61;

12 12 14 19 26 58 60 55;

14 13 16 24 40 57 69 56;

14 17 22 29 51 87 80 62;

18 22 37 56 68 109 103 77;

24 35 55 64 81 104 113 92;

49 64 78 87 103 121 120 101;

72 92 95 98 112 100 103 99];

QC = [17 18 24 47 99 99 99 99;

18 21 26 66 99 99 99 99;

24 26 56 99 99 99 99 99;

47 66 99 99 99 99 99 99;

99 99 99 99 99 99 99 99;

99 99 99 99 99 99 99 99;

99 99 99 99 99 99 99 99;

99 99 99 99 99 99 99 99];