**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная работа №3

**Случайные процессы в обработке изображений**

Вариант 2

**Выполнили:**

Крученков Евгений Андреевич

Кендысь Алексей Максимович,

студенты 4 курса, 7 группы,

специальность

“прикладная математика”

**Преподаватель:**

Старший преподаватель

кафедры ТВиМС ФПМИ,

Л.А. Хаткевич

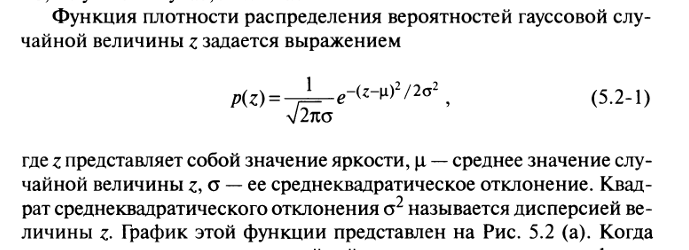
Минск, 2023

**Теоретические сведения**

Шум — это бессмысленная информация, добавляемая к данным, которая приводит к их повреждению или искажению. Шум позволяет затруднить распознавание содержимого изображения. В большинстве случаев шумы распространяются случайным образом. Соответственно, шум может обладать статистическими свойствами (распределение, дисперсия и др.). В процессе передачи изображения могут искажаться помехами, возникающими в каналах связи. Т.е. реальные данные часто содержат некоторое количество шума. Соответственно, важно научиться выдерживать шум и изучить его основные закономерности, т.к. модель должна быть способна обобщать зашумлённые обучающие данные.

Гауссовский шум является одной из самых простых и понятных разновидностей шумов. Он имеет хорошо понятные свойства и полностью определяется своим средним значением и стандартным отклонением, над ним можно добиться хорошего контроля и предсказуемости. Большинство шумов, встречающихся в природе, напоминают гауссовский шум, что делает его идеальным выбором для моделирования. Однако, его простота иногда оказывается столь привлекательной, что зачастую гауссовские модели используются даже в тех случаях, когда их применение оправдано, в лучшем случае, лишь частично.

Самый простой способ добавить гауссовский шум — создать гауссовское распределение такого же размера, как и изображение, а затем добавить его к исходному изображению.



**Выполнение задания**

**Подключение библиотек и задание параметров.**

import math

import matplotlib.pyplot as plt

import numpy as np

size = [500, 500, 3]

triangle\_side\_length = 60

distance\_between\_triangles = 15

triangle\_height = triangle\_side\_length

mu = 0

var1 = 0.1

var2 = 0.25

var3 = 0.5

var4 = 5

**Создание фона изображения.**

image = np.ndarray(size, dtype=int)

for i in range(size[0]):

    for j in range(size[1]):

        image[i][j][0] = 250

        image[i][j][1] = 230

        image[i][j][2] = 0

**Генерация треугольников.**

def in\_triangle(x1, y1, x2, y2, x3, y3, xp, yp):

    c1 = (x2-x1)\*(yp-y1)-(y2-y1)\*(xp-x1)

    c2 = (x3-x2)\*(yp-y2)-(y3-y2)\*(xp-x2)

    c3 = (x1-x3)\*(yp-y3)-(y1-y3)\*(xp-x3)

    return (c1<0 and c2<0 and c3<0) or (c1>0 and c2>0 and c3>0)

def get\_first\_triangle\_points(triangle\_side\_length, triangle\_height):

    x1 = 0

    y1 = 0

    x2 = triangle\_side\_length - 1

    y2 = 0

    x3 = math.ceil(triangle\_side\_length / 2) - 1

    y3 = triangle\_height - 1

    return x1, y1, x2, y2, x3, y3

def get\_second\_triangle\_points(triangle\_side\_length, triangle\_height):

    x1 = 0

    y1 = triangle\_height - 1

    x2 = math.ceil(triangle\_side\_length / 2) - 1

    y2 = 0

    x3 = triangle\_side\_length

    y3 = triangle\_height - 1

    return x1, y1, x2, y2, x3, y3

period\_x\_left\_triangles = triangle\_side\_length + (distance\_between\_triangles \* 2) + 1

period\_y\_left\_triangles = 2 \* triangle\_height + 2 \* distance\_between\_triangles

for i in range(size[0]):

    for j in range(size[1]):

        #first triangle

        x0, y0 = 0, 0

        x1, y1, x2, y2, x3, y3 = get\_first\_triangle\_points(triangle\_side\_length, triangle\_height)

        if (in\_triangle(x1, y1, x2, y2, x3, y3, (j - x0) % period\_x\_left\_triangles, (i - y0) % period\_y\_left\_triangles)):

            image[i][j][0] = 0

            image[i][j][1] = 180

            image[i][j][2] = 0

        #second triangle

        x0, y0 = math.ceil(triangle\_side\_length / 2) + distance\_between\_triangles, 0

        x1, y1, x2, y2, x3, y3 = get\_second\_triangle\_points(triangle\_side\_length, triangle\_height)

        if (in\_triangle(x1, y1, x2, y2, x3, y3, (j - x0) % period\_x\_left\_triangles, (i - y0) % period\_y\_left\_triangles)):

            image[i][j][0] = 0

            image[i][j][1] = 180

            image[i][j][2] = 0

        #third triangle

        x0, y0 = 0, triangle\_height + distance\_between\_triangles

        x1, y1, x2, y2, x3, y3 = get\_second\_triangle\_points(triangle\_side\_length, triangle\_height)

        if (in\_triangle(x1, y1, x2, y2, x3, y3, (j - x0) % period\_x\_left\_triangles, (i - y0) % period\_y\_left\_triangles)):

            image[i][j][0] = 0

            image[i][j][1] = 180

            image[i][j][2] = 0

        #fourth triangle

        x0, y0 = math.ceil(triangle\_side\_length / 2) + distance\_between\_triangles, triangle\_height + distance\_between\_triangles

        x1, y1, x2, y2, x3, y3 = get\_first\_triangle\_points(triangle\_side\_length, triangle\_height)

        if (in\_triangle(x1, y1, x2, y2, x3, y3, (j - x0) % period\_x\_left\_triangles, (i - y0) % period\_y\_left\_triangles)):

            image[i][j][0] = 0

            image[i][j][1] = 180

            image[i][j][2] = 0

**Вывод изображения.**

def show\_image(ax, image, title=None, cmap=None):

    ax.imshow(image, cmap=cmap)

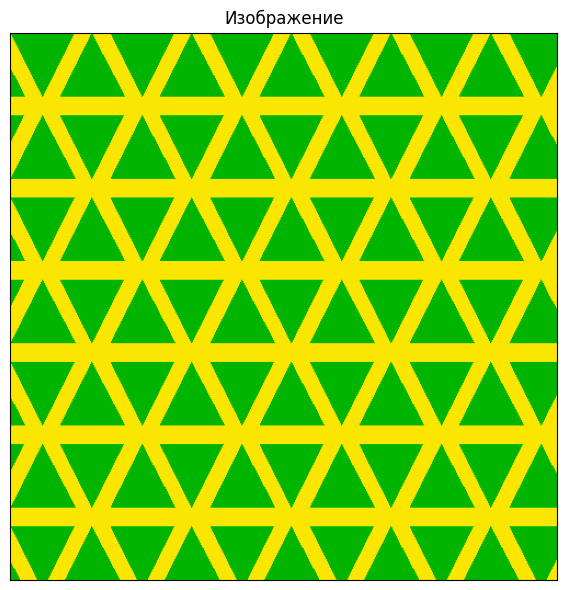
    ax.set\_title(title)

    ax.xaxis.set\_visible(False)

    ax.yaxis.set\_visible(False)

fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, image, title="Изображение", cmap="gray")



**Гауссовский шум.**

noise1 = np.random.normal(mu, math.sqrt(var1), size)

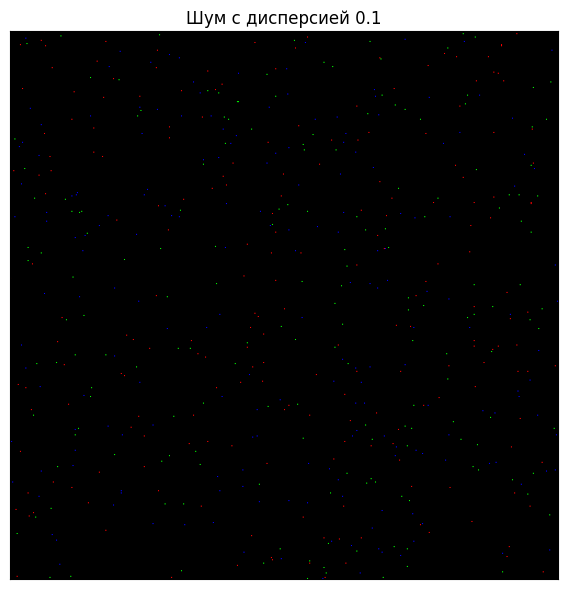
noise2 = np.random.normal(mu, math.sqrt(var2), size)

noise3 = np.random.normal(mu, math.sqrt(var3), size)

noise4 = np.random.normal(mu, math.sqrt(var4), size)

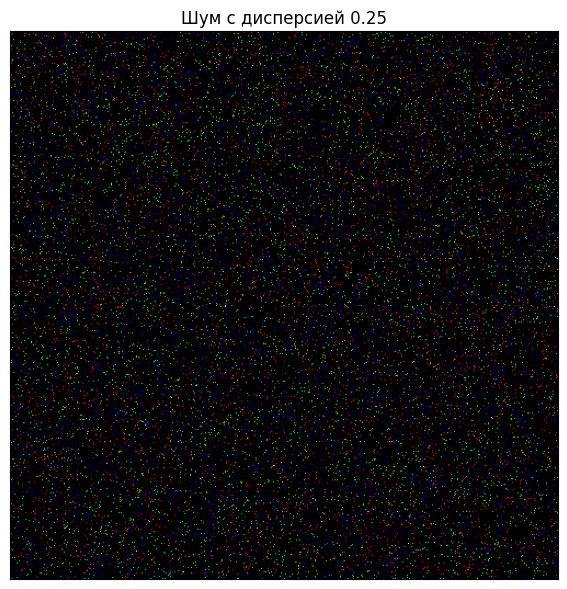
fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, noise1.astype(np.uint8), title="Шум с дисперсией " + str(var1), cmap="gray")



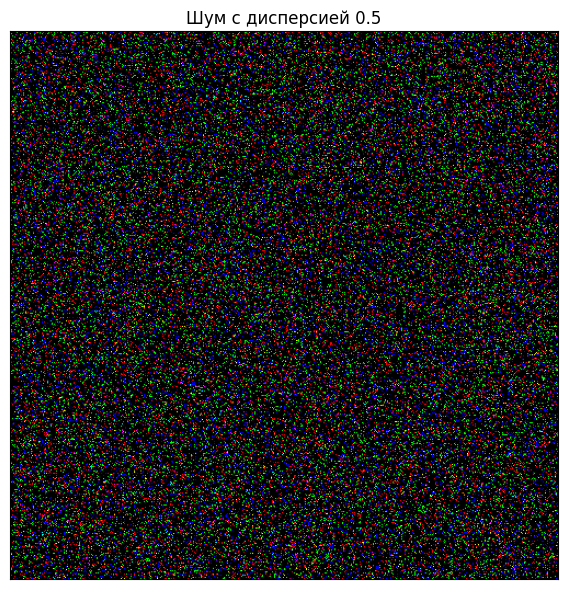
fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, noise2.astype(np.uint8), title="Шум с дисперсией " + str(var2), cmap="gray")



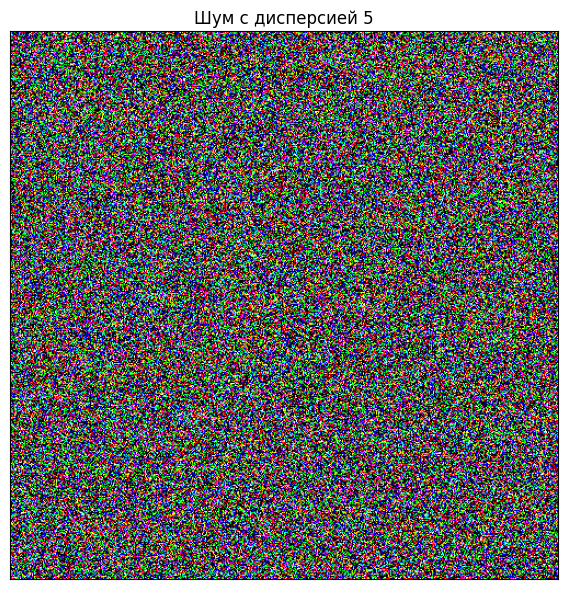
fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, noise3.astype(np.uint8), title="Шум с дисперсией " + str(var3), cmap="gray")



fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, noise4.astype(np.uint8), title="Шум с дисперсией " + str(var4), cmap="gray")



**Применение шума к изображению.**

image\_noised1 = image + noise1

image\_noised1 = image\_noised1.astype(np.uint8)

image\_noised2 = image + noise2

image\_noised2 = image\_noised2.astype(np.uint8)

image\_noised3 = image + noise3

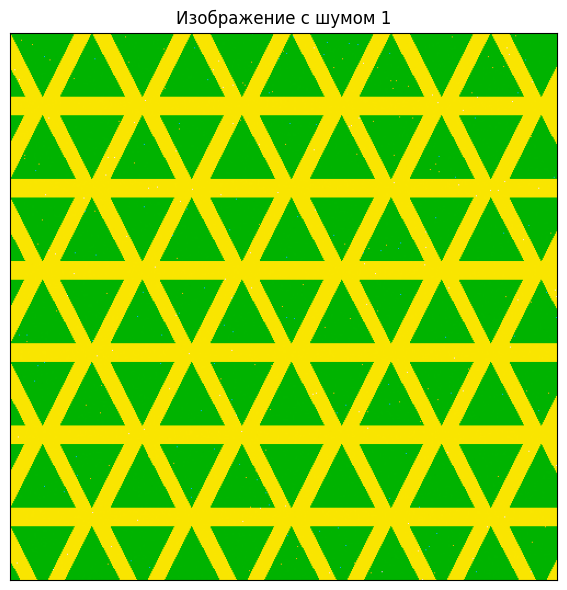
image\_noised3 = image\_noised3.astype(np.uint8)

image\_noised4 = image + noise4

image\_noised4 = image\_noised4.astype(np.uint8)

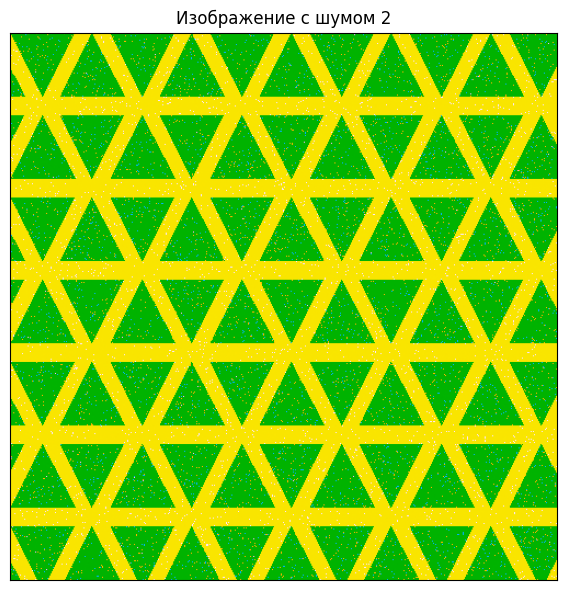
fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, image\_noised1, title="Изображение с шумом 1", cmap="gray")



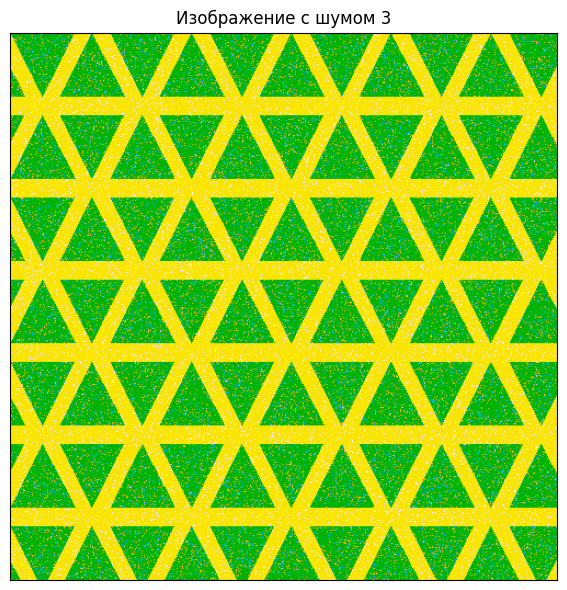
fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, image\_noised2, title="Изображение с шумом 2", cmap="gray")



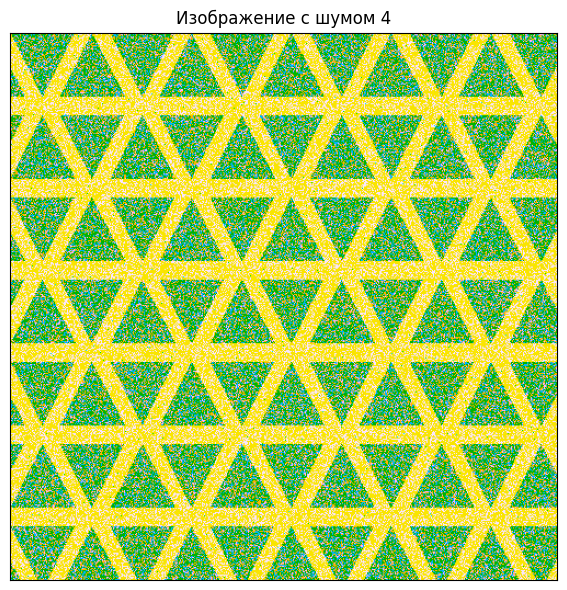
fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, image\_noised3, title="Изображение с шумом 3", cmap="gray")



fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, image\_noised4, title="Изображение с шумом 4", cmap="gray")



**Выводы**

При моделировании треугольников можно заметить закономерность, что упрощает задачу. Для проверки принадлежности точки треугольнику используется параметрическое задание треугольников через три прямые. Таким образом, можно проверить, принадлежит ли точка треугольнику, проходя по всем пикселям изображения, а также проверять эту принадлежность с некоторым периодом, т.к. изображение треугольников начинает повторяться. Моделирование было сделано таким образом, что можно задать любую длину основания треугольника, любую высоту треугольника, а также любое расстояние между треугольниками.

Что касается шума, то можно заметить, что с увеличением дисперсии растёт и зашумлённость исходного изображения. Так, с дисперсией 0.1 шум едва заметен, а с ростом дисперсии он начинает проявляться всё более явно, при дисперсии 5 изображение заметно портится. Это имеет смысл, т.к. при увеличении дисперсии растёт разброс значений в гауссовском распределении. Также стоит отметить, что шум получился цветным, т.к. генерировался для всех трёх цветовых каналов.