ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | К. А. Кочин |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «Моделирование на ЭВМ случайных величин RAND и GAUSS» |
| по курсу: Прикладная теория вероятностей и статистика |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4332 |  |  |  | А. А. Лютов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

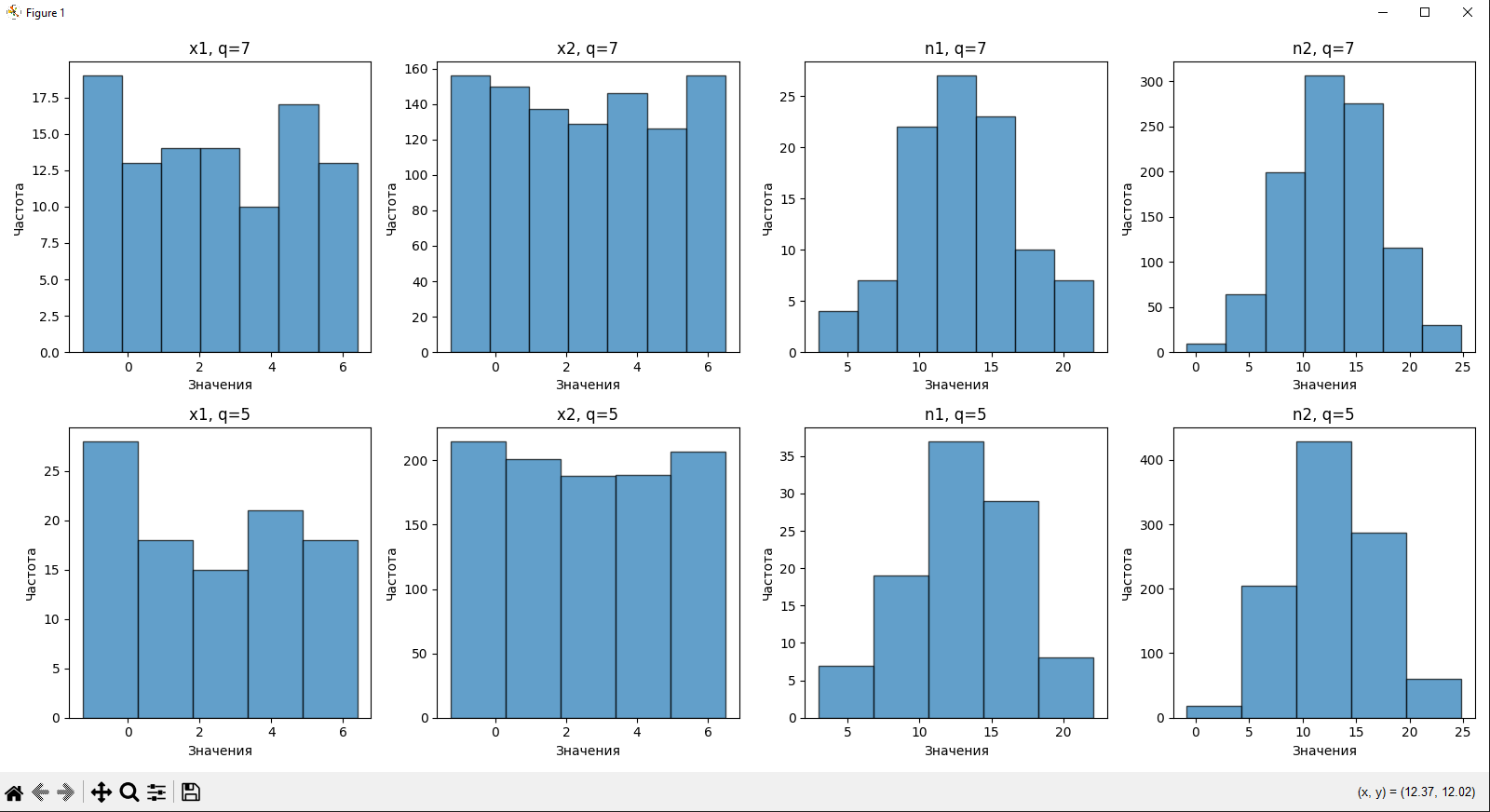
Санкт-Петербург 2025

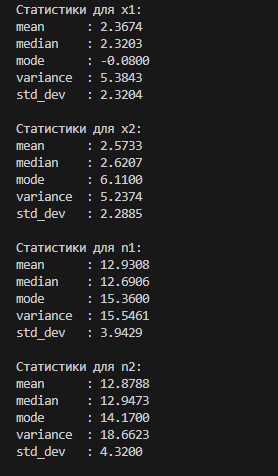
**Задание:**

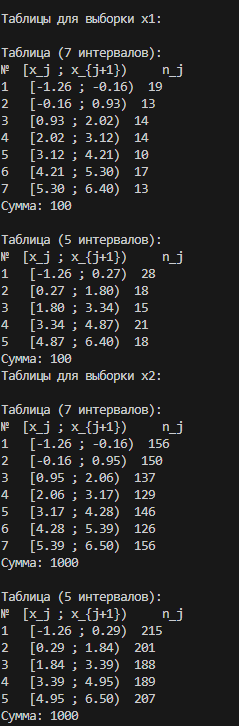
1. Сгенерировать реализацию случайной величины с равномерным законом распределения . Путем преобразований получить из нее реализацию случайной величины, распределенной по равномерному закону , (где = 1, 2, … – номер варианта). Данным способом получить две реализации с объемом выборки  и и сохранить их в файле для последующей обработки. Для каждой выборки построить две гистограммы: с 5-ю и 7-ю интервалами. Вычислить основные статистические характеристики (математическое ожидание, мода, медиана, дисперсия, среднее квадратическое отклонение) для обоих выборок.
2. Сгенерировать реализацию случайной величины с равномерным законом распределения . Путем преобразований получить из нее реализацию случайной величины, распределенной по нормальному закону , (где – номер варианта). Данным способом получить две реализации с объемом выборки и и сохранить их в файле для последующей обработки. Для каждой выборки построить две гистограммы: с 5-ю и 7-ю интервалами. Вычислить основные статистические характеристики (математическое ожидание, мода, медиана, дисперсия, среднее квадратическое отклонение) для обоих выборок.
3. Проверить гипотезу о нормальности распределения для каждой гистограммы из предыдущих пунктов (итого проверить 8 гипотез) с помощью критерия согласия χ2 при заданном уровне значимости , где  – номер варианта, % – операция взятия остатка при целочисленном делении.

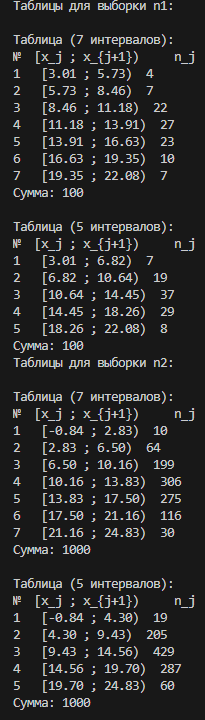
N = 13

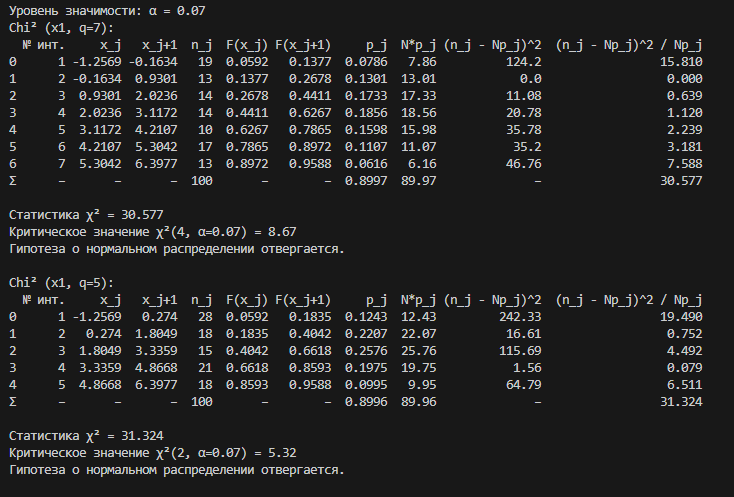
**Расчеты и графики:**

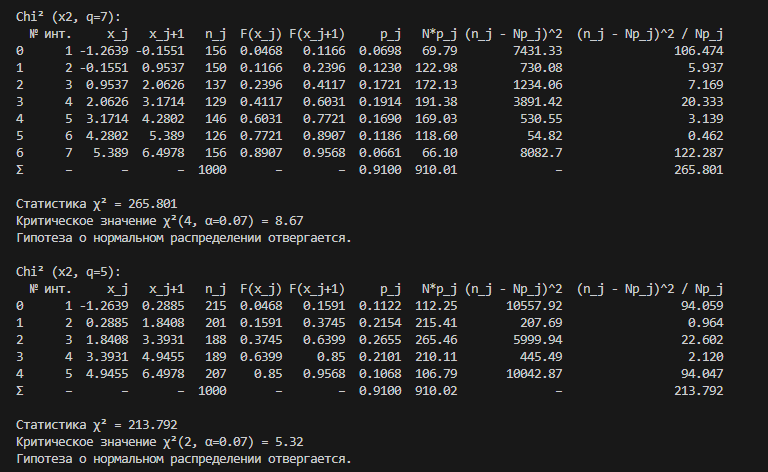


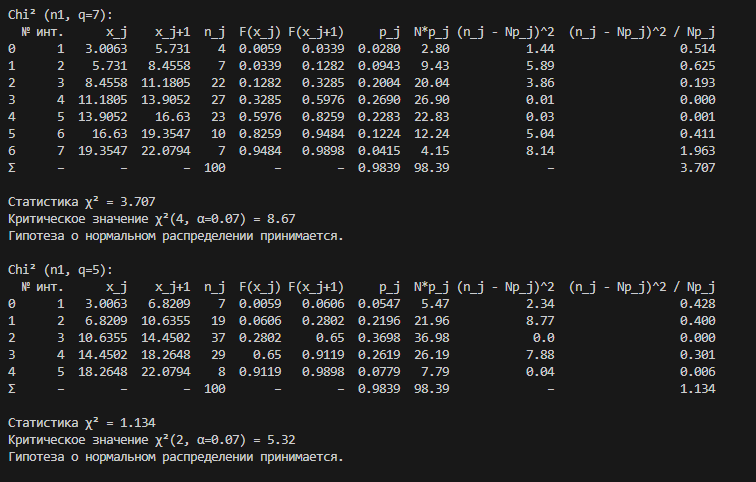


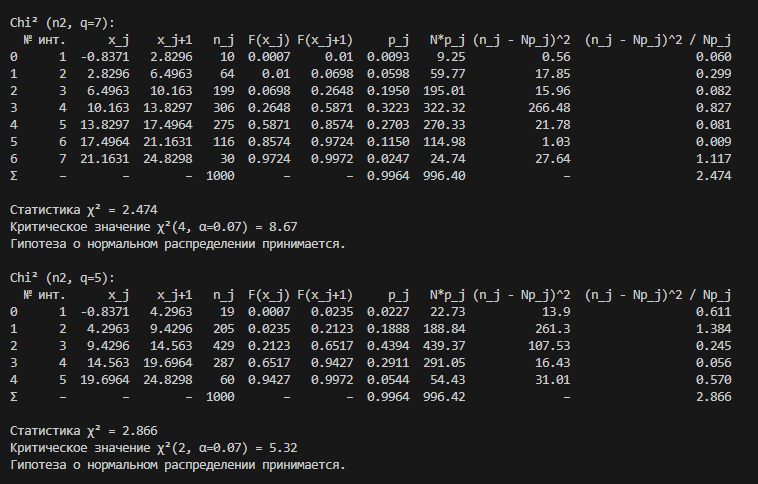












**Код программы:**

Lab3.py

import numpy as np

import matplotlib.pyplot as plt

import json

import pandas as pd

from scipy.stats import norm, chi2

from collections import Counter

N = 13

a, b = -N / 10, N / 2

M\_n, sigma\_n = N, N / 3

l1, l2 = 100, 1000

file\_path = "random\_samples.json"

alpha = 0.07

def GenerateNormalBoxMuller(size):

    adjusted\_size = size + (size % 2)

    u1, u2 = np.random.rand(adjusted\_size // 2), np.random.rand(adjusted\_size // 2)

    z1 = np.sqrt(-2 \* np.log(u1)) \* np.cos(2 \* np.pi \* u2)

    z2 = np.sqrt(-2 \* np.log(u1)) \* np.sin(2 \* np.pi \* u2)

    return np.concatenate([z1, z2])[:size]

try:

    with open(file\_path, "r") as f:

        data = json.load(f)

    x1, x2 = np.array(data["x1"]), np.array(data["x2"])

    n1, n2 = np.array(data["n1"]), np.array(data["n2"])

except (FileNotFoundError, KeyError, json.JSONDecodeError):

    np.random.seed(42)

    x1 = a + (b - a) \* np.random.rand(l1)

    x2 = a + (b - a) \* np.random.rand(l2)

    n1 = M\_n + sigma\_n \* GenerateNormalBoxMuller(l1)

    n2 = M\_n + sigma\_n \* GenerateNormalBoxMuller(l2)

    with open(file\_path, "w") as f:

        json.dump({

            "x1": x1.tolist(), "x2": x2.tolist(),

            "n1": n1.tolist(), "n2": n2.tolist(),

            "parameters": {"N": N, "a": a, "b": b, "M\_n": M\_n, "sigma\_n": sigma\_n}

        }, f, indent=4)

def ComputeStatistics(sample):

    values = np.round(sample, 2)

    mode = Counter(values).most\_common(1)[0][0]

    return {

        "mean": np.mean(sample),

        "median": np.median(sample),

        "mode": float(mode),

        "variance": np.var(sample, ddof=1),

        "std\_dev": np.std(sample, ddof=1)

    }

def PrintStatistics(name, stats):

    print(f"\nСтатистики для {name}:")

    for key, value in stats.items():

        print(f"{key:10}: {value:.4f}")

def ComputeHistogram(sample, q):

    R = np.ptp(sample)

    delta = R / q

    bins = np.linspace(np.min(sample), np.max(sample), q + 1)

    hist, \_ = np.histogram(sample, bins=bins)

    print(f"\nТаблица ({q} интервалов):")

    print("№  [x\_j ; x\_{j+1})     n\_j")

    for i in range(q):

        print(f"{i+1:<3} [{bins[i]:.2f} ; {bins[i+1]:.2f})  {hist[i]}")

    print(f"Сумма: {sum(hist)}")

    return hist, bins

def ChiSquareTable(sample, bins\_count=7, alpha=0.07):

    N = len(sample)

    mean = np.mean(sample)

    std = np.std(sample, ddof=1)

    hist, bin\_edges = np.histogram(sample, bins=bins\_count)

    intervals = list(zip(bin\_edges[:-1], bin\_edges[1:]))

    results, chi2\_stat = [], 0

    for j, (xj, xj1) in enumerate(intervals):

        Fj, Fj1 = norm.cdf(xj, loc=mean, scale=std), norm.cdf(xj1, loc=mean, scale=std)

        pj, npj, nj = Fj1 - Fj, N \* (Fj1 - Fj), hist[j]

        diff\_sq = (nj - npj)\*\*2

        chi\_contrib = diff\_sq / npj if npj > 0 else 0

        chi2\_stat += chi\_contrib

        results.append({

            "№ инт.": j + 1, "x\_j": round(xj, 4), "x\_j+1": round(xj1, 4),

            "n\_j": nj, "F(x\_j)": round(Fj, 4), "F(x\_j+1)": round(Fj1, 4),

            "p\_j": round(pj, 4), "N\*p\_j": round(npj, 2),

            "(n\_j - Np\_j)^2": round(diff\_sq, 2),

            "(n\_j - Np\_j)^2 / Np\_j": round(chi\_contrib, 3)

        })

    S, q = 2, bins\_count

    k = q - S - 1

    chi2\_critical = chi2.ppf(1 - alpha, df=k)

    hypothesis\_result = "принимается" if chi2\_stat <= chi2\_critical else "отвергается"

    df = pd.DataFrame(results)

    df.loc["Σ"] = ["–", "–", "–", N, "–", "–",

                   round(sum(r["p\_j"] for r in results), 4),

                   round(sum(r["N\*p\_j"] for r in results), 2), "–", round(chi2\_stat, 3)]

    print(df)

    print(f"\nСтатистика χ² = {chi2\_stat:.3f}")

    print(f"Критическое значение χ²({k}, α={alpha}) = {chi2\_critical:.2f}")

    print(f"Гипотеза о нормальном распределении {hypothesis\_result}.\n")

    return df

def PlotHistograms():

    fig, axes = plt.subplots(2, 4, figsize=(16, 8))

    samples = [x1, x2, n1, n2]

    labels = ["x1", "x2", "n1", "n2"]

    for i, (sample, label) in enumerate(zip(samples, labels)):

        for j, q in enumerate([7, 5]):

            axes[j, i].hist(sample, bins=q, edgecolor='black', alpha=0.7)

            axes[j, i].set\_title(f"{label}, q={q}")

            axes[j, i].set\_xlabel("Значения")

            axes[j, i].set\_ylabel("Частота")

    plt.tight\_layout()

    plt.savefig("histograms.png")

    plt.show()

for name, sample in zip(["x1", "x2", "n1", "n2"], [x1, x2, n1, n2]):

    PrintStatistics(name, ComputeStatistics(sample))

print("\nТаблицы для выборки x1:")

ComputeHistogram(x1, 7)

ComputeHistogram(x1, 5)

print("Таблицы для выборки x2:")

ComputeHistogram(x2, 7)

ComputeHistogram(x2, 5)

print("Таблицы для выборки n1:")

ComputeHistogram(n1, 7)

ComputeHistogram(n1, 5)

print("Таблицы для выборки n2:")

ComputeHistogram(n2, 7)

ComputeHistogram(n2, 5)

PlotHistograms()

print(f"\nУровень значимости: α = {alpha:.2f}")

for label, sample in zip(["x1", "x2", "n1", "n2"], [x1, x2, n1, n2]):

    for q in [7, 5]:

        print(f"Chi² ({label}, q={q}):")

        ChiSquareTable(sample, q)

**Вывод:**

В результате выполнения лабораторной работы я создал четыре выборки с равномерным и нормальным распределением, описал их характеристики и проверил гипотезу о нормальности распределения.