МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО

ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (МОСКОВСКИЙ ПОЛИТЕХ)

**«Отчёт по выполненной лабораторной работе №7. Статистические методы исследования точности»**

Студент: \_\_\_\_\_\_\_\_\_\_\_\_ / Рыжов Артем Алексеевич, 221-324 /   
  *подпись*  *ФИО, группа*

Студент: \_\_\_\_\_\_\_\_\_\_\_\_ / Пермяков Глеб Сергеевич, 221-324 /   
  *подпись*  *ФИО, группа*

Студент: \_\_\_\_\_\_\_\_\_\_\_\_ / Фильчинков Даниил Павлович, 221-324 /   
  *подпись*  *ФИО, группа*

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_ / Толстиков Антон Витальевич, К.Т.Н. /

*подпись ФИО, уч. звание и степень*

Москва, 2025

**ВВЕДЕНИЕ**

Целью данной работы было создать приложение, которое использует статистические методы исследования точности для нахождения процента годных деталей, процента исправимого и неисправимого брака. Программа строит схему расположения поля рассеивания размеров относительного поля допуска и предлагает направление смещения поля рассеивания размеров относительного поля допуска для снижения или устранения неисправимого брака. Пользователь сможет настроить точность расчёта. Программа написана на языке программирования Python 3.12 с использованием библиотеки matplotlib для отображения графических элементов.

**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

Плотность вероятности или дифференциальная функция распределения случайной величины непрерывного типа, подчиняющейся закону нормального распределения, имеет следующее выражение (рисунок 1). Где x – переменная случайная величина; φ(x) – плотность вероятности; σ – среднее квадратическое отклонение случайной величины x от ¬x; ¬x – среднее значение (математическое ожидание) величин x; e – основание натуральных логарифмов.

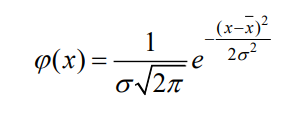


Рисунок 1 – Формула плотности вероятности

Формула вычисляет значение интеграла от 0 до z функции плотности вероятности стандартного нормального распределения (рисунок 2).

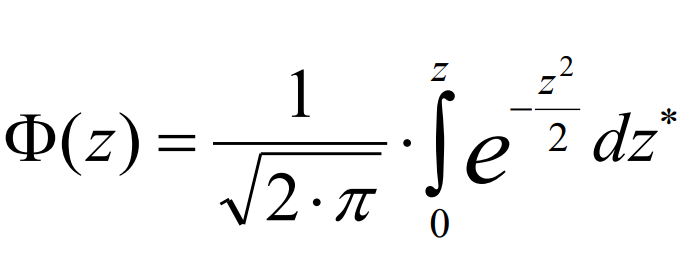


Рисунок 2 – Интеграл функции плотности вероятности стандартного нормального распределения

**ПРАКТИЧЕСКАЯ ЧАСТЬ**

1. **Реализация статистических методов**

Реализуем вычисление плотности вероятности (рисунок 1) с помощью отдельной функции (листинг 1). Функция принимает на вход значение переменной «x», которая используется для определения точки, в которой необходимо рассчитать значение плотности вероятности. Переменная nx - математическое ожидание, o - стандартное отклонение. При нормальном стандартном распределении математическое ожидание равно нулю, а стандартное отклонение равно единице.

Листинг 1 – Функция плотности вероятности

def normal\_distribution(x: float, nx: float = 0, o: float = 1):return (1 / (o \* math.sqrt(2 \* math.pi))) \* math.exp(-0.5 \* ((x - nx) / o) \*\* 2)

Добавляем реализацию вычисления значения интеграла от 0 до b функции плотности стандартного нормального распределения (листинг 2). На вход функция принимает два параметра: b – конечная точка интегрирования, n – точность вычисления. Реализация функции написана по формуле (рисунок 2).

Листинг 2 – Функция интеграла функции плотности

def F(b: float, n: int = 10000):a = 0 *# начальная точка интегрирования* dx = (b - a) / n *# ширина каждого разбиения (шаг интегрирования).* sum = 0 *# переменная для накопления суммы значений функции на каждом шаге.* for i in range(n):  
 xi = a + i \* dx *# вычисление текущей точки на интервале интегрирования.* sum += normal\_distribution(xi) \* dx *# добавление площади прямоугольника с высотой func(xi) и шириной dx к общей сумме.* return sum

Реализуем подсчёт количества годных деталей и брака в одной функции (листинг 3). На вход функция принимает 5 значений: ei – нижнее предельное отклонение, es – верхнее предельное отклонение, nx – наладочный размер, o – среднее квадратичное отклонение, n – точность вычислений. Функция возвращает кортеж из значений, характеризующих количество годных деталей, количество неисправимого брака, количество исправимого брака.

Листинг 3 – Функция подсчёта количества годных деталей и брака

def calculate(ei: float, es: float, nx: float, o: float, n: int = 10000) -> Tuple[float, float, float]: *# Определение количества годных деталей* t2: float = (es - nx) / o  
 t1: float = (ei - nx) / o  
 p\_suitable\_parts: float = round((F(t2, n) - F(t1, n)) \* 100, 2) *# Определение неисправимого брака* t2: float = (ei - nx) / o  
 t1: float = (nx - 3 \* o - nx) / o  
 p\_incorrigible\_marriage: float = round((F(t2, n) - F(t1, n)) \* 100, 2) *# Определение исправимого брака* t2: float = (nx + 3 \* o - nx) / o  
 t1: float = (es - nx) / o  
 p\_fixable\_marriage: float = round((F(t2, n) - F(t1, n)) \* 100, 2)return p\_suitable\_parts, p\_incorrigible\_marriage, p\_fixable\_marriage

1. **Реализация интерфейса**

Импортируем модули для работы с графиками и данными (листинг 4).

Листинг 4 – Импорт модулей

import matplotlib.pyplot as plt  
from matplotlib.widgets import Slider, Button, TextBox  
import numpy as np

Создаём функцию для построения графика нормального распределения (листинг 5). Функция принимает на вход 3 параметра: ax - объект класса Axes, представляющий собой область для построения графика, nx – наладочный размер, o – среднее квадратичное отклонение. Функция возвращает 3 значения: x - диапазон значений вокруг среднего nx в пределах ±3 стандартных отклонений, y - содержит значения функции плотности вероятности для каждого xi, line - объект линии, используемый для дальнейшего изменения или настройки графика

Листинг 5 – Данные для построения графика

def build\_graph(ax: plt.Axes, nx: float, o: float) -> Tuple[np.ndarray, List[float], plt.Line2D]:x: np.ndarray = np.linspace(nx - 3 \* o, nx + 3 \* o, 1000)  
 y: List[float] = [normal\_distribution(xi, nx, o) for xi in x]  
 (line,) = ax.plot(x, y, label="Нормальное распределение")  
 return x, y, line

Реализуем функцию закрашивания областей годных деталей и брака (листинг 6). Функция принимает 5 параметров: ax - объект класса Axes, представляющий собой область для построения графика, ei - нижнее предельное отклонение, es - верхнее предельное отклонение, nx - наладочный размер, o - среднее квадратичное отклонение.

Листинг 6 – Закрашивание областей

def fill\_areas(ax: plt.Axes, ei: float, es: float, nx: float, o: float) -> None:x\_suitable = np.linspace(ei, es, 100)  
 y\_suitable = [normal\_distribution(xi, nx, o) for xi in x\_suitable]  
 ax.fill\_between(  
 x\_suitable, y\_suitable, color="green", alpha=0.3, label="Годные детали"  
 )  
  
 x\_incorrigible = np.linspace(nx - 4 \* o, ei, 100)  
 y\_incorrigible = [normal\_distribution(xi, nx, o) for xi in x\_incorrigible]  
 ax.fill\_between(  
 x\_incorrigible,  
 y\_incorrigible,  
 color="red",  
 alpha=0.3,  
 label="Неисправимый брак",  
 )  
  
 x\_fixable = np.linspace(es, nx + 4 \* o, 100)  
 y\_fixable = [normal\_distribution(xi, nx, o) for xi in x\_fixable]  
 ax.fill\_between(  
 x\_fixable, y\_fixable, color="orange", alpha=0.3, label="Исправимый брак"  
 )

Реализуем функцию, которая добавляет текст с информацией о браке (листинг 7). Параметры функции suitable\_parts, incorrigible\_marriage и fixable\_marriage отвечают за процент годных деталей, неисправимого и исправимого брака соответственно.

Листинг 7 – Функция добавления текста с информацией о браках

def add\_text(  
 ax: plt.Axes,  
 nx: float,  
 o: float,  
 y: List[float],  
 suitable\_parts: float,  
 incorrigible\_marriage: float,  
 fixable\_marriage: float,  
):ax.text(  
 nx - 3.5 \* o,  
 0.8 \* max(y),  
 f"Годные детали: {suitable\_parts}%\n"  
 f"Неисправимый брак: {incorrigible\_marriage}%\n"  
 f"Исправимый брак: {fixable\_marriage}%",  
 bbox=dict(facecolor="white", alpha=0.8),  
 )

После реализации предыдущих функций получаем график, отображающий все необходимые данные о годных деталях и браках (рисунок 3).

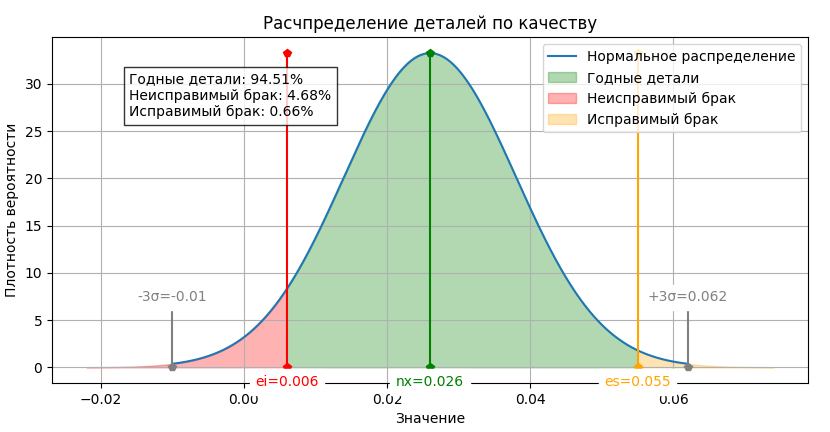


Рисунок 3 – График распределения деталей по качеству

Добавляем функцию, которая перерисовывает график (листинг 8). Функция обновления реагирует на изменение параметров слайдеров (листинг 9), которые создаются под графиком (рисунок 4).

Листинг 8 – Функция обновления графика

def update(val):  
 ei = ei\_slider.val  
 es = es\_slider.val  
 nx = nx\_slider.val  
 o = o\_slider.val  
 accuracy: int = int(accuracy\_text\_box.text)  
 if (accuracy == 0):  
 accuracy = initial\_accuracy  
 x = np.linspace(nx - 3 \* o, nx + 3 \* o, 1000)  
 y = [normal\_distribution(xi, nx, o) for xi in x]  
 line.set\_data(x, y)  
  
 ax.clear()  
 ax.plot(x, y, lw=2, label="Нормальное распределение")  
 suitable\_parts, incorrigible\_marriage, fixable\_marriage = calculate(ei, es, nx, o, accuracy)  
 fill\_areas(ax, ei, es, nx, o)  
 draw\_lines(ei, es, nx, o, True)  
 add\_text(ax, nx, o, y, suitable\_parts, incorrigible\_marriage, fixable\_marriage)  
 ax.set\_xlabel("Значение")  
 ax.set\_ylabel("Плотность вероятности")  
 ax.set\_title("Распределение деталей по качеству")  
 ax.legend()  
 ax.grid(True)  
 fig.canvas.draw\_idle()

Листинг 9 – Привязка слайдеров к функции обновления

ei\_slider = Slider(ax\_ei, "ниж. предельное отклонение ei", 0.001, 0.08, valinit=ei)  
es\_slider = Slider(ax\_es, "верх. предельное отклонение es", 0.01, 0.08, valinit=es)  
nx\_slider = Slider(ax\_nx, "наладочный размер nx", 0.01, 0.06, valinit=nx)  
o\_slider = Slider(ax\_o, "среднее квадратичное откл. o", 0.001, 0.06, valinit=o)  
  
*# Привязка функции обновления к слайдерам*ei\_slider.on\_changed(update)  
es\_slider.on\_changed(update)  
nx\_slider.on\_changed(update)  
o\_slider.on\_changed(update)

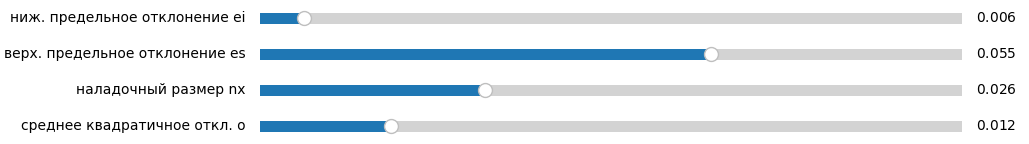


Рисунок 4 – Отображение слайдеров

Разрабатываем текстовое поле, которое предлагает рекомендации по изменению наладочного размера с целью устранения неисправимого брака (листинг 10).

Листинг 10 – Рекомендации по смещению

*# Рекомендации по смещению наладочного размера*

recommendation\_ax = fig.add\_axes((0.4, 0.025, 0.07, 0.04))  
recommendation\_tb = TextBox(recommendation\_ax, "Рекомендуется сместить наладочный размер на ", initial=str(round(ei - (nx - 3 \* o), 3)))

*# Автоматическая поправка с помощью кнопки*btn\_rec\_box = fig.add\_axes((0.5, 0.025, 0.1, 0.04))  
btn\_recommendation = Button(btn\_rec\_box, "Fix", hovercolor="0.975")  
btn\_recommendation.on\_clicked(fix\_recommendation)

Создаём функцию запуска оконного приложения (листинг 11), в которую добавляем реализованные функции в виде вложенных. Также создаём новые элементы, такие как: кнопка сброса слайдеров до значений по умолчанию, поля ввода для фиксации значений слайдеров по умолчанию, кнопка сброса полей фиксаций. Также добавляем поле ввода точности.

Листинг 11 – Функция запуска оконного приложения

def start\_ui() -> None:

ei: float = 0.006 *# 0.002*es: float = 0.055 *# 0.035*nx: float = 0.026 *# 0.014*o: float = 0.012 *# 0.009*initial\_accuracy: int = 1000

…

plt.show()

В результате получаем оконное приложение со всем функционалом, позволяющим отслеживать годные детали, исправимые и неисправимые браки. Также приложение позволяет изменять стартовые значения для анализа изменения графика и получать рекомендации по изменению наладочного размера (рисунок 5).

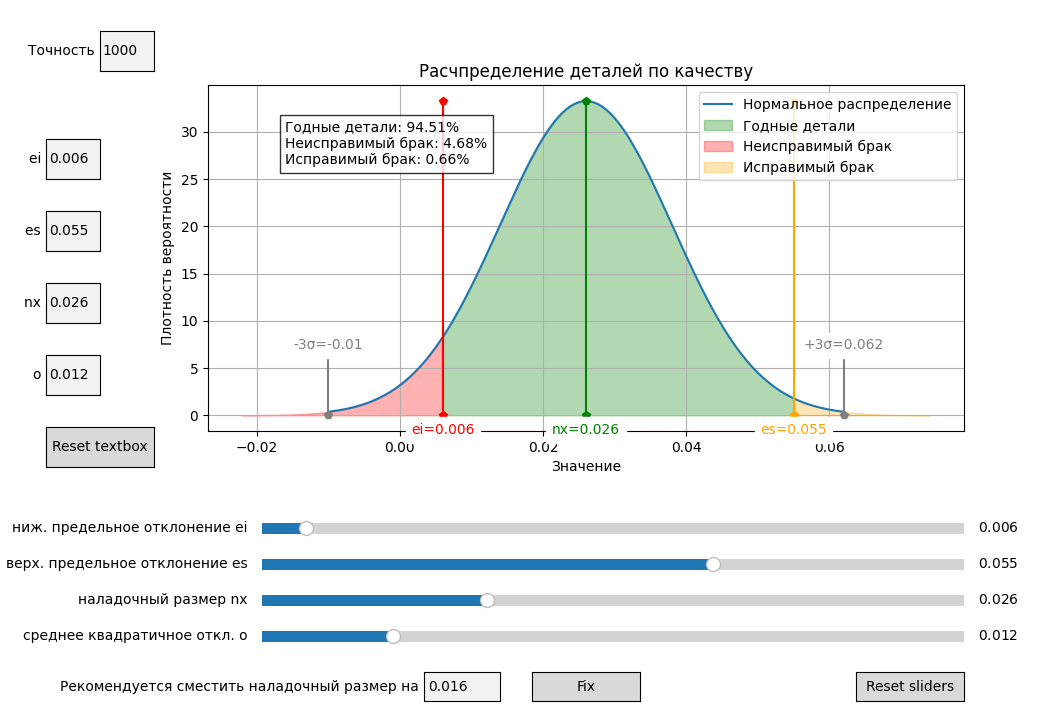


Рисунок 5 – Финальная версия приложения

**ВЫВОД**

Мы создали оконное приложение, которое использует статистические методы для анализа точности и определения процента годных деталей, процента исправимого и неисправимого брака. Программа способна решать задачи на основе входных параметров и оптимизировать значение наладочного размера. Для удобства пользователя в приложении представлены график и слайдеры, которые позволяют анализировать поведение функции в зависимости от изменения исходных параметров.