ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

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

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «Реализация на ЭВМ алгоритмов оценивания среднего риска» |
| по курсу: Прикладная теория вероятностей и статистика |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

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

Санкт-Петербург 2025

**Задание:**

1. Заполнить таблицу 2 в соответствии с вариантом *N* при известных значениях параметров α и β:

|  |  |
| --- | --- |
| , , |  |

где  - номер варианта.

Таблица 2. Параметры состояния КА при известных значениях параметров α и β

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Состояние | S0 | | S1 | |
| Априорная  вероятность | (КА исправен) | | (КА неисправен) | |
| Принятое  решение | Н00 | Н10 | Н01 | Н11 |
| Оценка  решения | правильное | ошибка 1-го рода | ошибка 2-го рода | правильное |
| Вероятность решения | 1 – α | α | β | 1 – β |
| Стоимость  потерь | C00 | C10 | C01 | C11 |
| Средний  риск |  |  |  |  |

* + Определить вероятность ошибочного решения оператора при известных априорных вероятностях и
  + Определить средний риск (потери), связанный с ошибочными решениями оператора, при условии, что известны априорные вероятности и состояния КА и матрица потерь.

1. Определить граничное значение температуры *х*гр, выше которого техническое состояние КА считается неисправным, а также вероятность ложной тревоги *P*(*H*01), вероятность пропуска дефекта *P*(*H*10) и средний риск *R*:
   * по методу минимального среднего риска;
   * по методу минимальной вероятности ошибочного решения.

**Вариант:**

| № варианта | Значение параметра | | | |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| 13 | 23 | 3,5 | 35 | 5,2 |

**Листинг:**

import numpy as np

import scipy.stats as stats

N = 13

alpha = N / 100

beta = N / 200

P0 = 0.97

P1 = 0.03

x0\_bar = 23

sigma0 = 3.5

x1\_bar = 35

sigma1 = 5.2

C00 = 0

C11 = 0

C01 = 20 \* (N + 5) / N

C10 = (N + 5) / N

print("\nПараметры состояния КА при известных значениях параметров alpha и beta:")

print(f"alpha = {alpha:.3f}, beta = {beta:.3f}")

print("\nВероятности решений:")

print(f"P(H00) = {1 - alpha:.3f}")

print(f"P(H10) = {alpha:.3f}")

print(f"P(H01) = {beta:.3f}")

print(f"P(H11) = {1 - beta:.3f}")

print("\nПотери:")

print(f"C00 = {C00}, C10 = {C10:.3f}, C01 = {C01:.3f}, C11 = {C11}")

R00 = P0 \* C00 \* (1 - alpha)

R10 = P0 \* C10 \* alpha

R01 = P1 \* C01 \* beta

R11 = P1 \* C11 \* (1 - beta)

total\_risk = R00 + R10 + R01 + R11

print("\nСредний риск по компонентам:")

print(f"R00 = {R00:.4f}, R10 = {R10:.4f}, R01 = {R01:.4f}, R11 = {R11:.4f}")

print(f"Общий средний риск: {total\_risk:.4f}")

def bayes\_risk\_threshold(x0, sigma0, x1, sigma1, P0, P1, C10, C01):

    k = (P1 \* (C01 - C11)) / (P0 \* (C10 - C00))

    a = 1/(2\*sigma1\*\*2) - 1/(2\*sigma0\*\*2)

    b = x0/sigma0\*\*2 - x1/sigma1\*\*2

    c = (x1\*\*2)/(2\*sigma1\*\*2) - (x0\*\*2)/(2\*sigma0\*\*2) - np.log((sigma0/sigma1)\*k)

    discriminant = b\*\*2 - 4\*a\*c

    if discriminant < 0:

        return None

    x1 = (-b + np.sqrt(discriminant)) / (2\*a)

    x2 = (-b - np.sqrt(discriminant)) / (2\*a)

    if x0 < x1 < x1\_bar:

        return x1

    elif x0 < x2 < x1\_bar:

        return x2

    else:

        return (x0 + x1\_bar)/2

def min\_error\_threshold(x0, sigma0, x1, sigma1, P0, P1):

    k = P1/P0

    a = 1/(2\*sigma1\*\*2) - 1/(2\*sigma0\*\*2)

    b = x0/sigma0\*\*2 - x1/sigma1\*\*2

    c = (x1\*\*2)/(2\*sigma1\*\*2) - (x0\*\*2)/(2\*sigma0\*\*2) - np.log((sigma0/sigma1)\*k)

    discriminant = b\*\*2 - 4\*a\*c

    if discriminant < 0:

        return None

    x1 = (-b + np.sqrt(discriminant)) / (2\*a)

    x2 = (-b - np.sqrt(discriminant)) / (2\*a)

    if x0 < x1 < x1\_bar:

        return x1

    elif x0 < x2 < x1\_bar:

        return x2

    else:

        return (x0 + x1\_bar)/2

x\_risk = bayes\_risk\_threshold(x0\_bar, sigma0, x1\_bar, sigma1, P0, P1, C10, C01)

if x\_risk is not None:

    alpha\_risk = 1 - stats.norm.cdf(x\_risk, loc=x0\_bar, scale=sigma0)

    beta\_risk = stats.norm.cdf(x\_risk, loc=x1\_bar, scale=sigma1)

    R\_risk = P0 \* C10 \* alpha\_risk + P1 \* C01 \* beta\_risk

else:

    alpha\_risk, beta\_risk, R\_risk = None, None, None

x\_error = min\_error\_threshold(x0\_bar, sigma0, x1\_bar, sigma1, P0, P1)

if x\_error is not None:

    alpha\_error = 1 - stats.norm.cdf(x\_error, loc=x0\_bar, scale=sigma0)

    beta\_error = stats.norm.cdf(x\_error, loc=x1\_bar, scale=sigma1)

    R\_error = P0 \* C10 \* alpha\_error + P1 \* C01 \* beta\_error

else:

    alpha\_error, beta\_error, R\_error = None, None, None

print("\nОпределение граничного значения температуры xгр и среднего риска R:")

if x\_risk is not None:

    print("Метод: Минимального среднего риска")

    print(f"  Граница температуры: {x\_risk:.2f}")

    print(f"  Вероятность ложной тревоги (α): {alpha\_risk:.4f}")

    print(f"  Вероятность пропуска дефекта (β): {beta\_risk:.4f}")

    print(f"  Средний риск: {R\_risk:.4f}")

else:

    print("Метод минимального среднего риска: не удалось найти действительное граничное значение")

if x\_error is not None:

    print("\nМетод: Минимальной вероятности ошибки")

    print(f"  Граница температуры: {x\_error:.2f}")

    print(f"  Вероятность ложной тревоги (α): {alpha\_error:.4f}")

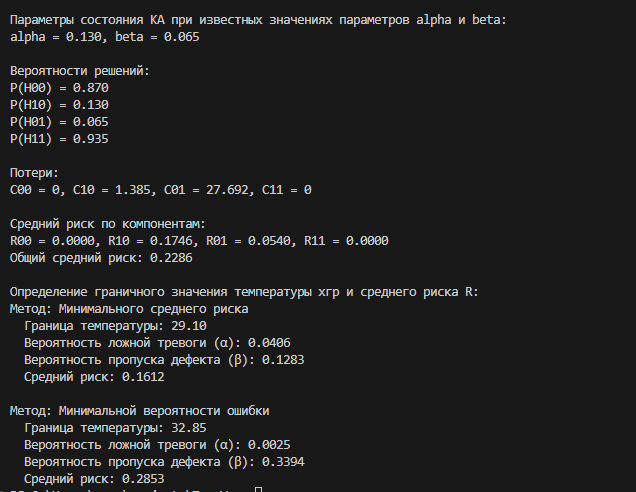
    print(f"  Вероятность пропуска дефекта (β): {beta\_error:.4f}")

    print(f"  Средний риск: {R\_error:.4f}")

else:

    print("\nМетод минимальной вероятности ошибки: не удалось найти действительное граничное значение")

**Расчеты и таблицы:**



**Вывод:** В результате выполнения данной работы я смог реализовать алгоритмы нахождения среднего риска на ЭВМ, такие как метод минимального среднего риска и метод минимальной вероятности ошибочного решения.